INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Diabetic-Friendly Multi-Agent Recommendation System for Restaurants based on Social Media Sentiment Analysis and Multi-Criteria Decision Making

BRUNO CÉSAR JANTARADA TEIXEIRA
julho de 2022

Diabetic-Friendly Multi-Agent Recommendation System for Restaurants based on Social Media Sentiment Analysis and Multi-Criteria Decision Making

Bruno César Jantarada Teixeira

Student ID: 1020504

Dissertation to obtain a Master's Degree in Artificial Intelligence

Advisor: Dr. Maria Goreti Carvalho Marreiros, Coordinator Teacher at Instituto Superior de Engenharia do Instituto Politécnico do Porto, Portugal

Co-advisor: Diogo Martinho, Instituto Superior de Engenharia do Instituto Politécnico do Porto, Portugal

Jury

President

Dr. António Constantino Lopes Martins, Assistant Teacher at Instituto Superior de Engenharia do Instituto Politécnico do Porto, Portugal

Vowels:

Dr. Victor Manuel Rodrigues Alves, Auxiliar Teacher at Universidade do Minho, Portugal

Dr. Maria Goreti Carvalho Marreiros, Coordinator Teacher at Instituto Superior de Engenharia do Instituto Politécnico do Porto, Portugal

Porto, June 2022

Abstract

Lifestyle, poor diet, stress, among other factors, strongly contribute to aggravate people's health problems, such as diabetes and high blood pressure. Some of these problems could be avoided if some of the essential recommendations for the practice of a healthy lifestyle were followed. This dissertation proposes a solution designed improve the quality of life of diabetic patients, more specifically in the context of finding restaurant in the nearby location, that are more suitable for the health needs of this patients. A diabetic-friendly feature that will combine multi-criteria decision making built through Multi-Agent System (MAS) that considers the user preferences initially recorded, and that provides the user with three category recommendations to potentially benefit and improve the user lifestyle and health. The solution proposes the use of Case-Based Reasoning algorithm to enable the solution to evolve and improve in each interaction with the user. Sentiment Analysis was also used for identifying the restaurant reviews score, since this is one of the defined criteria for the solution.

Keywords: Recommender System, Diabetic-friendly, Multi-Agent System, Sentiment Analysis, Muti-Criteria Decision Making, Case-based Reasoning, k-Nearest Neighbors.

Resumo

Estilo de vida, má alimentação, stress, entre outros fatores, contribuem fortemente para agravar os problemas de saúde das pessoas como diabetes e hipertensão. Alguns desses problemas poderiam ser evitados se algumas das recomendações essenciais para a prática de um estilo de vida saudável fossem seguidas. Esta dissertação propõe uma solução concebida para melhorar a qualidade de vida dos doentes diabéticos, mais concretamente no contexto da procura de restaurantes, nas proximidades, que sejam mais adequados às necessidades de saúde destes doentes. Uma funcionalidade "diabetic-friendly" que combinará a tomada de decisão multicritério construída através de um Sistema Multi Agente que considera as preferências do utilizador registradas inicialmente e fornecerá ao usuário três recomendações de categorias que potencialmente beneficiam o estilo de vida e a saúde. A solução propõe o uso do algoritmo de Raciocínio Baseado em Casos para permitir que a solução evolua e melhore a cada interação com o utilizador. A Análise Sentimental também foi utilizada para identificar a pontuação das avaliações do restaurante, pois este é um dos critérios definidos para a solução.

Palavras-chave: Sistema de Recomendação, Diabetic-friendly, Sistema Multi Agente, Análise Sentimental, Tomada de Decisão Multi-Critério, Raciocínio Baseado em Casos.

Dedication

This dissertation is dedicated to my parents and my sister for always being present and support all my decisions in life. In particular to my sister, since she is a Type 1 diabetic, and all the proposed work was done always with her in my mind. I would like to give my special thanks to both my advisor Dr. Maria Goreti Marreiros and co-advisor Diogo Martinho, who have given me the support and the clarification needed in the more difficult times. I would like to give my appreciation also to my company partner, for helping me managing all the daily work, when I was in need for time to focus for the current work. Last but not least, to my girlfriend who had the patience and love to keep me in track on the not so good days.

Thank you all.

Index

Figure Index

Table Index

Acronyms

List of Acronyms

ACP American College of Physicians

AHP Analytical Hierarchical Processing

ACL Agent Communication Language

API Application Programming Interface

BOWs Bag of Words

CBR Case-Based Reasoning

CAP Cognitive Analytic Process

DM Diabetes mellitus

ELECTRE Elimination Et Choix Traduisant la Realité

FIPA Foundation for Intelligent Physical Agents

GDSS Group Decision Support Systems

IDDM Insulin-dependent diabetes mellitus

JADE Java Agent Development

KIF Knowledge Interchange Format

KQML Knowledge Query Manipulation Language

MAS Multi-Agent System

MCRS Multi-criteria Recommender Systems

MCDM Multi-criteria Decision-Making

NIDDM Non-insulin-dependent diabetes mellitus

OM Opinion Mining

PC Public Communication

PrC Private Communication

RSs Recommender Systems

REST Representational State Transfer

SA Sentiment Analysis

TOPSIS Technique for Order Performance by Similarity to Ideal Solution

VAS Visual Analogue Scale

WSM Weighted Sum Model

WWW World Wide Web

WAN Wide Area Network

1 Introduction

1.1 Context

According with World Health Organization, the number of people with diabetes rose from 108 million in 1980 to 422 million in 2014. Prevalence has increased faster in low- and middle-income countries than in high-income countries. Diabetes is a major cause of blindness, kidney failure, heart attacks, stroke, and lower limb amputation. Between 2000 and 2016, there was a 5% increase in premature mortality from diabetes. In 2019, nearly 1.5 million deaths were directly caused by diabetes. Another 2.2 million deaths were attributed to high blood glucose in 2012. A healthy diet, regular physical activity, maintaining a normal body weight, and avoiding tobacco use are ways to prevent or delay the onset of type 2 diabetes. Diabetes can be treated, and its consequences avoided or delayed with diet, physical activity, medication and regular checkups and treatment for complications [1].

1.2 Motivation

Just in Portugal alone there are currently more than 1 million people identified with diabetes [2]. Diabetes is among the 10 leading causes of death worldwide. Lifestyle, inadequate diet, stress, among other factors, strongly contribute to aggravate the diabetic patient health condition. Some problems can be avoided if some of the essential recommendations for the practice of a healthy lifestyle are followed. The purpose of this dissertation is to tackle some of the potential ways to prevent complications to diabetic patients, by creating a diabetic-friendly framework solution that has the potential to be integrated in multiple solutions.

1.3 Problem

Diabetes is characterized by high blood sugar level over prolonged period. If left untreated, diabetes causes many health complications. Acute complications can include hyperosmolar hyperglycemic state, diabetic ketoacidosis, or even death. Serious long-term health complications include chronic kidney disease, foot ulcers, damage to the eyes, cardiovascular disease, and stroke. Diabetes occurs due to either the inability of the pancreas to produce

enough insulin, or the body cells improperly responding to the insulin produced. Type 1, Type 2, and the Gestational diabetes are the three major types of diabetes mellitus, although there is a collection of other specific types. The Type 1 diabetes results from the failure of pancreas to produce enough insulin due to the loss of beta cells caused by an autoimmune response. Type 2 diabetes begins with the insulin resistance, a condition in which the cells fail to properly respond to insulin. As the disease keeps progressing, a lack of insulin may develop. A combination of insufficient exercise and excessive body weight is the most common cause. Gestational diabetes is third major form and occurs when a pregnant woman without previous history of diabetes develops high blood sugar levels. Adequate dieting, with good nutrition and regular exercise are very important in preventing or controlling diabetes. Prevention and treatment of diabetes involve maintaining healthy diet, regular physical exercise, normal body weight, and also avoiding use of tobacco. Maintaining a healthy diet, low-fat diet, low-calorie diet, palaeolithic diet, very low carbohydrate diet, raw foodyism, and/or ketogenic diet can help prevent or manage diabetes [3]. The use of gamification techniques to support elderly people has proven to be beneficial to improve wellbeing as well as both physical, cognitive, social, and emotional state of the elderly person [4] and this group of people is extremely relevant for this disease.

1.4 Objectives

This dissertation proposes the development of a recommender system that allow diabetic patients take better decisions when they are looking for restaurant delivery food, more adjusted to their disease requirements. The system cloud be part of a dedicated mobile application or easily integrated with already in the market applications like Uber Eats, Zomato or TripAdvisor in the form of functionality. The system will be responsible for collecting user preferences through an initial survey. After this initial setup and based on the user location (GPS coordinates), the system should be able to deliver restaurant recommendations by three main categories (Best Choice, Nearby and Reviews). The system will use a set of artificial intelligence techniques to deliver best recommendations and evolve along with user interactions in the long term, always with the main goal to increase diabetic patient healthy diet.

1.5 Expected Results

The main goal for the proposed solution is to develop an agent based framework that should allow the user to receive restaurant recommendations that potentially improve their healthy habits and diet. This framework based on multi-agent systems allow the agent that represents the user to receive recommendations, after internal negotiations between all the agents in the system. Those agents will behave in a collaborative style in the negotiation process, always focusing on providing the best restaurant recommendations possible.

After all the research and theoretical analysis which will be presented in the next chapters with the identification of concepts and definitions to be incorporated within the proposed framework, the development took place. At the end were performed two use cases to validate and verify the impact of all the assumptions previously assumed.

By this, the following hypothesis for study in this dissertation was defined by:

- The use of the CBR algorithm will provide more accurate restaurant recommendations, compared with those provided without the use of the CBR algorithm.

1.6 Structure of the Dissertation

In the section will be presented the structure of this dissertation, being mentioned as follow and overview on the next chapters of this document.

In chapter 2 will be exposed the main concepts required to understand the proposed solution. This should give the reader a base of knowledge that allow him/her to understand the approach taken in the implementation process. There are also presented related works from the literature review in line with the proposed solution, where different techniques were used, which clear some initial thoughts about mechanisms with advantages and disadvantages of each chosen option.

Chapter 3 will present the solution modeling proposed for the work presented in this dissertation where are presented the solution architecture, domain model diagrams as well as tools and methodologies applied.

In chapter 4 all the implementation process is described, as well as some particularities of the proposed solution and decisions made during the process.

At the end, in chapter 5, are presented the results and the discussion achieved when the defined hypothesis was applied and tested. Finishing with chapter 6 where conclusions about the work are done as well as identified ideas for future work as well as the scientific impact of the proposed solution.

2 Context

In this chapter will be provide information about the main topics and concepts applied in this dissertation. They should provide the reader with the required knowledge to understand the proposed solution.

2.1 Diabetes

Diabetes mellitus (DM), commonly called diabetes, is a group of metabolic disorders and diseases characterized by high blood sugar levels over prolonged period. The symptoms of high blood glucose (sugar) levels include increased thirst, increased hunger, and frequent urination [5]. Diabetes can cause many complications if not carefully treated and controlled. So far, there is no known cure for diabetes. Acute complications can include hyperosmolar hyperglycemic state, diabetic ketoacidosis, or death [5]. Serious long-term complications are cardiovascular disease, stroke, foot ulcers, damage to the eyes, and chronic kidney disease. Diabetes is as a result of either pancreas not making enough insulin, or cells of the body improperly responding to the insulin produced [6]. There are three main types of diabetes, and a collection of “other specific types”. Diabetes type 1 results from failure of pancreas to produce enough insulin caused by loss of beta cells. Diabetes type 1 was earlier referred to as the juvenile diabetes or insulin-dependent diabetes mellitus (IDDM). The loss of beta cells is as a result of an autoimmune response [7]. The cause of the autoimmune response is unknown [5]. Type 2 diabetes begins with the insulin resistance, a condition where cells fail to properly respond to insulin. As the disease progresses, lack of insulin may develop. This form was previously known as "adult-onset diabetes" or "non-insulin-dependent diabetes mellitus" (NIDDM). The most common cause is combination of excessive body weight (obesity) and insufficient exercise [5]. Gestational diabetes is the third most common form, and occurs when a pregnant woman without previous history of diabetes mellitus develops high blood sugar levels [5]. Prevention and treatment of diabetes involve maintaining healthy diet, regular physical exercise, normal body weight, and also avoiding tobacco use. Control of blood pressure, eye care, and maintaining proper foot care are important for individuals with the disease [5]. Type 1 diabetes must be properly managed with insulin injections. The type 2 diabetes may be treated with medication with or without insulin. The Insulin and some oral medication can cause low blood sugar [8]. Weight loss surgery in individuals with obesity is sometimes very effective measure in people with type 2 diabetes. Gestational diabetes often resolves after birth of the baby [9]. Diabetic patients are advised to avoid sugars and sugary soft drinks. They may instead opt for diet soft drinks sweetened with sugar substitutes such as sugar alcohols [10] which contribute little or no-calorie. In 2017, an estimated 425 million individuals had diabetes worldwide [11], with type 2 diabetes making up around 90% of the cases [12]. This represents 8.8% of adult population, with equal rates in both men and women. Trends suggest that the rates will continue to rise [11]. Diabetes at least doubles an individual's risk of early death. In 2017 alone, diabetes resulted in about 3.2 to 5.0 million deaths [11]. The global economic costs of diabetes-related health expenditures in 2017 were estimated at US$727 billion [11]. Average medical expenditures among individuals with diabetes are around 2.3 times higher [13].

2.1.1 Preventions of Diabetes

There preventive measure for type 1 diabetes is not known [5]. Type 2 diabetes—which accounts for 85 to 90% of all cases worldwide—can normally be prevented or delayed by engaging in physical activity, eating a healthy diet, and maintaining normal body weight [5]. Higher levels of physical activity (over 90 minutes per day) reduce the risks of diabetes by 28% [14]. Effective dietary changes known to help prevent diabetes include maintaining diet rich in fiber and whole grain, and choosing good fats, such as polyunsaturated fats found in vegetable oils, nuts, and fish [15]. Eating less red meat and other foods with saturated fat, in addition to limiting sugary beverages can also help prevent diabetes. The smoking of tobacco is also associated with increased risk of diabetes and its related complications, thus smoking cessation can be a key preventive measure as well [16]. The relationship between type 2 and the main modifiable risk factors (unhealthy diet, physical inactivity, tobacco use, and excess weight) is similar in all the regions of the world. In addition, there is increasing evidence that the fundamental determinants of diabetes mellitus are a reflection of the main forces driving social, cultural, and economic change: globalization, population aging, urbanization, and the general health policy environment [15].

2.1.2 Diabetes Management

The management of diabetes focuses on keeping blood sugar levels very close to normal, without causing a low blood sugar level. This can often be accomplished with dietary changes, weight loss, exercise, and use of appropriate medications (oral medications, insulin). Learning about the disease and also actively participating in the treatment is very important, since complications are usually less severe and far less common in individuals who have well-managed their blood sugar levels [17]. Per the American College of Physicians (ACP), the goal of the treatment is attaining an HbA1C level of 7 to 8% [18]. Attention is also given to other health issues that may accelerate negative effects of diabetes, including smoking, high blood pressure, lack of regular exercise, and metabolic syndrome obesity. Specialized footwear is used widely to reduce the risks of ulcers in at-risk diabetic feet, though evidence for the efficiency of this remains equivocal [19].

Those with diabetes can benefit from the education about the disease and its treatment, dietary changes required, and exercise, with the aim of keeping both the short-term and the long-term blood glucose levels within adequate and acceptable bounds. Additionally, given the associated higher risk of cardiovascular disease, modifications of lifestyle are recommended to control blood pressure [20] [21], including healthy eating, regular exercise, and maintaining normal weight (BMI 18 to 25). Weight loss can prevent progression from the prediabetes to type 2 diabetes, result in a partial remission in those with diabetes, or decrease the risks of cardiovascular disease [22] [23]. No single dietary pattern is the best for all individuals with diabetes [24]. Healthy dietary patterns, such as Mediterranean diet, low-carbohydrate diet DASH diet, are usually recommended, although no evidence support one over the others [22] [23]. According to the ADA, reducing the overall carbohydrate intake for people with diabetes has shown the most evidence for glycemic improvement, and for those with type 2 diabetes who are unable to meet the glycemic target or where reducing anti-glycemic medication is a priority, very-low or low carbohydrate diets are viable approach [23]. For overweight individuals with diabetes type 2, a diet that achieves loss of weight is effective [24].

2.2 Food Friend

The proposed work has been presented within the Food Friend project [25]. Food Friend is focused on two aspects of diet-related disease: malnutrition prevention for patients who require tube feeding and nutritional transmural care for chronic diseases such as diabetes. This project count with the collaboration of 17 partners from 5 different countries (Portugal, Netherlands, Belgium, Turkey and Czech Republic).

Within the national context the Portuguese use case is related with the management of patients with Type 2 diabetes diseases comprising topics related to diet, physical activity, and other factors that can alter the quality of life of these patients. The major outcome will be a quantified self, evidence-based coaching solution for self-monitoring and management of diabetes type II disease based on food intake and lifestyle monitoring. The transparent integration of this type of technology in the daily routine of patients will improve self-monitoring disease, increase treatment adherence and the promotion of healthy behaviors, having as consequence diminishing the acute complications of diabetes.

2.3 Recommender Systems

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user [26]. The first challenge for designing a RS is to define the users and purpose of specific context or domain in a proper way [27]. For this dissertation the selected approach technique defined was Multi-Criteria Recommender Systems, since that is the more suitable to achieve the expected results, but there are other approaches like:

- Collaborative filtering;
- Content-based filtering;
- Session-based recommender systems
- Reinforcement learning for recommender systems
- Risk-aware recommender systems
- Mobile recommender systems
- Hybrid recommender systems
- Association rule mining

2.3.1 Collaborative Filtering

One approach to the design of recommender systems that has wide use is collaborative filtering [28]. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. The system generates recommendations using only information about rating profiles for different

users or items. By locating peer users/items with a rating history similar to the current user or item, they generate recommendations using this neighborhood. Collaborative filtering methods are classified as memory-based and model-based. A well-known example of memory-based approaches is the user-based algorithm [29], while that of model-based approaches is the Kernel-Mapping Recommender [30]. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Many algorithms have been used in measuring user similarity or item similarity in recommender systems. For example, the k-nearest neighbor (k-NN) approach [31] and the Pearson Correlation as first implemented by Allen [32].

When building a model from a user's behavior, a distinction is often made between explicit and implicit forms of data collection. Examples of explicit data collection include the following:

- Asking a user to rate an item on a sliding scale;
- Asking a user to search;
- Asking a user to rank a collection of items from favorite to least favorite;
- Presenting two items to a user and asking him/her to choose the better one of them;
- Asking a user to create a list of items that he/she.

Examples of implicit data collection include the following:

- Observing the items that a user views in an online store.
- Analyzing item/user viewing times [33];
- Keeping a record of the items that a user purchases online;
- Obtaining a list of items that a user has listened to or watched on his/her computer;
- Analyzing the user's social network and discovering similar likes and dislikes.

Collaborative filtering approaches often suffer from three problems: cold start, scalability, and sparsity [34].

Cold start: For a new user or item, there isn't enough data to make accurate recommendations. Note: one commonly implemented solution to this problem is the Multi-armed bandit algorithm [35] [36] [37] [38] [39].

Scalability: There are millions of users and products in many of the environments in which these systems make recommendations. Thus, a large amount of computation power is often necessary to calculate recommendations.

Sparsity: The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings.

One of the most famous examples of collaborative filtering is item-to-item collaborative filtering (people who buy x also buy y), an algorithm popularized by Amazon.com's recommender system [40]. Many social networks originally used collaborative filtering to recommend new friends, groups, and other social connections by examining the network of connections between a user and their friends [41]. Collaborative filtering is still used as part of hybrid systems.

2.3.2 Content-based filtering

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item and a profile of the user's preferences [42] [43]. These methods are best suited to situations where there is known data on an item (name, location, description, etc.), but not on the user. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on an item's features. In this system, keywords are used to describe the items, and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items similar to those that a user liked in the past or is examining in the present. It does not rely on a user sign-in mechanism to generate this often-temporary profile. In particular, various candidate items are compared with items previously rated by the user, and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research. To create a user profile, the system mostly focuses on two types of information:

1. A model of the user's preference

2. A history of the user's interaction with the recommender system.

Basically, these methods use an item profile (i.e., a set of discrete attributes and features) characterizing the item within the system. To abstract the features of the items in the system, an item presentation algorithm is applied. A widely used algorithm is the tf–idf representation (also called vector space representation) [44]. The system creates a content-based profile of users based on a weighted vector of item features. The weights denote the importance of each feature to the user and can be computed from individually rated content vectors using a variety of techniques. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks in order to estimate the probability that the user is going to like the item [45]. A key issue with content-based filtering is whether the system can learn user preferences from users' actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended. For example, recommending news articles based on news browsing is useful. Still, it would be much more useful when music, videos, products, discussions, etc., from different services, can be recommended based on news browsing. To overcome this, most content-based recommender systems now use some form of the hybrid system. Content-based recommender systems can also include opinion-based recommender systems. In some cases, users are allowed to leave text reviews or feedback on the items. These user-generated texts are implicit data for the recommender system because they are potentially rich resources of both feature/aspects of the item and users' evaluation/sentiment to the item. Features extracted from the user-

generated reviews are improved meta-data of items, because as they also reflect aspects of the item like meta-data, extracted features are widely concerned by the users. Sentiments extracted from the reviews can be seen as users' rating scores on the corresponding features. Popular approaches of opinion-based recommender system utilize various techniques including text mining, information retrieval, sentiment analysis and deep learning [46].

2.3.3 Session-based recommender systems

These recommender systems use the interactions of a user within a session [47] to generate recommendations. Session-based recommender systems are used at Youtube [48] and Amazon [49]. These are particularly useful when history (such as past clicks, purchases) of a user is not available or not relevant in the current user session. Domains where session-based recommendations are particularly relevant include video, e-commerce, travel, music and more. Most instances of session-based recommender systems rely on the sequence of recent interactions within a session without requiring any additional details (historical, demographic) of the user. Techniques for session-based recommendations are mainly based on generative sequential models such as Recurrent Neural Networks [47] [50], Transformers [51], and other deep learning based approaches [52] [53].

2.3.4 Reinforcement learning for recommender systems

The recommendation problem can be seen as a special instance of a reinforcement learning problem whereby the user is the environment upon which the agent, the recommendation system acts upon in order to receive a reward, for instance, a click or engagement by the user [48] [54] [55]. One aspect of reinforcement learning that is of particular use in the area of recommender systems is the fact that the models or policies can be learned by providing a reward to the recommendation agent. This is in contrast to traditional learning techniques which rely on supervised learning approaches that are less flexible, reinforcement learning recommendation techniques allow to potentially train models that can be optimized directly on metrics of engagement, and user interest [56].

2.3.5 Multi-criteria recommender systems

Multi-criteria recommender systems (MCRS) can be defined as recommender systems that incorporate preference information upon multiple criteria. Instead of developing recommendation techniques based on a single criterion value, the overall preference of user u for the item *i*, these systems try to predict a rating for unexplored items of *u* by exploiting preference information on multiple criteria that affect this overall preference value. Several researchers approach MCRS as a multi-criteria decision making (MCDM) problem, and apply MCDM methods and techniques to implement MCRS systems.[58]

2.3.6 Risk-aware recommender systems

The majority of existing approaches to recommender systems focus on recommending the most relevant content to users using contextual information, yet do not take into account the risk of disturbing the user with unwanted notifications. It is important to consider the risk of upsetting

the user by pushing recommendations in certain circumstances, for instance, during a professional meeting, early morning, or late at night. Therefore, the performance of the recommender system depends in part on the degree to which it has incorporated the risk into the recommendation process. One option to manage this issue is DRARS, a system which models the context-aware recommendation as a bandit problem. This system combines a content-based technique and a contextual bandit algorithm [57].

2.3.7 Mobile recommender systems

Mobile recommender systems make use of internet-accessing smart phones to offer personalized, context-sensitive recommendations. This is a particularly difficult area of research as mobile data is more complex than data that recommender systems often have to deal with. It is heterogeneous, noisy, requires spatial and temporal autocorrelation, and has validation and generality problems [58]. There are three factors that could affect the mobile recommender systems and the accuracy of prediction results: the context, the recommendation method and privacy [59]. Additionally, mobile recommender systems suffer from a transplantation problem – recommendations may not apply in all regions (for instance, it would be unwise to recommend a recipe in an area where all of the ingredients may not be available). One example of a mobile recommender system are the approaches taken by companies such as Uber and Lyft to generate driving routes for taxi drivers in a city [58]. This system uses GPS data of the routes that taxi drivers take while working, which includes location (latitude and longitude), time stamps, and operational status (with or without passengers). It uses this data to recommend a list of pickup points along a route, with the goal of optimizing occupancy times and profits.

2.3.8 Hybrid recommender systems

Most recommender systems now use a hybrid approach, combining collaborative filtering, content-based filtering, and other approaches. There is no reason why several different techniques of the same type could not be hybridized. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model. Several studies that empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrated that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem, as well as the knowledge engineering bottleneck in knowledge-based approaches [60]. Netflix is a good example of the use of hybrid recommender systems [61]. The website makes recommendations by comparing the watching and searching habits of similar users (i.e., collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering). Some hybridization techniques include:

- **Weighted** - Combining the score of different recommendation components numerically;

- **Switching** - Choosing among recommendation components and applying the selected one;

- **Mixed** - Recommendations from different recommenders are presented together to give the recommendation;

- **Feature Combination** - Features derived from different knowledge sources are combined together and given to a single recommendation algorithm [62];

- **Feature Augmentation** - Computing a feature or set of features, which is then part of the input to the next technique [62];

- **Cascade** - Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones;

- **Meta-level** - One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique [63].

2.4 Multi-Criteria Decision-Making

Multicriteria decision-making (MCDM) refers to making decisions in the presence of multiple and usually conflicting criteria. Fuzzy decision-making is used where vague and incomplete data exist for the solution. Fuzzy multicriteria decision-making is one of the most popular problems handled by the researchers in the literature [64]. The decision-making steps are usually interconnected and are often carried out in a completely informal way, being defined as:

- Identification of alternatives, decision hypotheses;

- Feasibility analysis of each of the alternatives:

    - Delete the ones that do not satisfy restrictions;

- Definition of comparison / comparison (eg cost).

When it is considered that a single criterion is sufficient to represent preferences of the decision-maker we are in the presence of a trivial decision problem, but if difficulties arise when the decision is based on several criteria, we usually have conflicting issues and is not possible to find an alternative that is better than the others in all criteria simultaneously. There is no methodology that allows the best alternative to be decided and only the decision agent's intervention will allow a conclusion to be reached, which will result in combination of his preferences over the criteria alternatives. In multi-criteria decision making are two main different types of problems:

- Multi-attribute problems;
- Multi-objective problems.

2.4.1 Multi-attribute problems

Multi-attribute problems are defined by:

- the number of alternatives is limited and the attributes you know explicitly;
- Main characteristics:
    - The alternatives are completely defined and assumed feasible;
    - Attributes may be determinist, probabilistic, fuzzy (or mixed);
    - The problem may be:
        - Choice -select the best alternative;
        - Ranking – draw a complete order of alternatives;
        - Sorting – select the best k alternatives from a list of n>=k.

2.4.2 Multi-objective problems

Multi-objective problems are defined by:

- A set of admissible solutions defined through a set of constraints;
- The objectives are explained through the objective functions;
- Main characteristics:
    - Alternatives are not known in advance;
    - Optimization procedures are usually needed;
    - May have a big number of constraints and decision variables;
    - May not be completely described by the mathematical formulation;
    - Sometimes interpreted as optimization problems with more than one objective function (vector optimization).

The identification of a complete and non-overlapping family of criteria is essential for a correct multicriteria approach and the decision support can consist of modelling the decision problem, identification of efficient solutions (mainly in multi-objective problems), structuring the set of

efficient solutions, in a non-imperative way and an interactive process until choose the preferred decision. The criteria can be grouped in families as follow:

- Exhaustive – nothing important is left out;
- Consistent – if two alternatives A and B are equivalent except in criterion k, and if in that criterion a_k is better than b_k, then alternative A must be considered globally;
- Non-redundant – if one of the criterion is deleted the two above conditions fail;
- Legible – the number of criteria should not be excessive;
- Operability – the family of criteria must be accepted by all the stakeholders and decision-makers.

Group Decision Support Systems (GDSS) are also used in ubiquitous contexts and more recently in web-based approaches [65]. Supporting decision-making processes when the elements of a group are geographically dispersed and on a tight schedule is a complex task [66]. Using styles of behaviour, from which decision-makers' intentions can be modelled into agents, are one option to solve the limitations in the decision-makers' interaction. The use of argumentation also allows to define an agent not only according to the preferences of the decisionmaker but also according to his interests towards the decision-making process [67] [68]. Multiple Criteria Decision Analysis Framework which include features like Consensus-Based Group Decision Support System, is used by autonomous software agents to support decision-makers according to their specific needs/interests and can greatly facilitates the definition of important procedures, allowing decision-makers to take advantage of deciding as a group and to understand the reasons behind the different recommendations and proposals [69].

2.4.3 Literature approaches and methods

There are at least four different approaches identified on the literature. The American School or School of the Multi-Attribute Utility Theory (MAUT) that focus on approaches that better catch decision maker preferences. The French School or European school focused on approaches where decision makers have less influence on decision, the interactive method or multi-objective mathematical programming models and the Cognitive Analytic Process (CAP).

2.4.3.1 ELECTRE family methods (Elimination Et Choix Traduisant la REalité)

The method was first proposed by Bernard Roy in 1985 and follows the Frech School approach. In this method comparing the pairs of alternatives should consider, indifference, preference, and veto. Outranking relations between pairs of alternatives for each criterion and aggregation process that leads to a certain structuring of alternatives are also characteristics of this method.

2.4.3.2 TOPSIS (Technique for Order Performance by Similarity to Ideal Solution)

The TOPSIS method is based on the concepts of positive ideal solution (PIS) and negative ideal solution (NIS) as points on a Euclidean space. The ranking of the alternatives is based on the relative distances of each alternative to the PIS and NIS points. The best alternative is the one that have the smallest distance to the PIS and biggest distance to the NIS. In order to achieve a

utility function is defined. The utility function will converts numerical attribute scales to utility unit scales, assign weights to each criteria and compare alternatives using the overall utility score.

2.4.3.3 AHP (Analytical Hierarchical Processing)

This method was initially proposed by Thomas Saaty, in the University of Pittsburgh in the 70's [70]. It belongs to the family of normative methods and tries to decompose the decision problem into a hierarchy of subproblems, performing elicitation of pairwise comparison judgments. The normal steps of the method are defined by:

- Problem structuring and definition of the hierarchy;
- Elicitation of pairwise comparisons:
    - Normalizing the resulting matrix;
- Derivation of priority vectors:
    - Averaging the values in each row to get the corresponding rating;
- Check consistency;
- Calculate overall scores.

2.4.3.4 CAP (Cognitive Analytic Process)

This method was proposed by João Carneiro, et al [71] and it includes cognitive aspects for Multiple Criteria Decision Analysis. The Cognitive Analytic Process allows decision-makers to represent their preferences about a problem (criteria and alternatives) and represent what is referred as their intentions, being directed towards supporting group decision-making. Decision-makers could express their intentions by selecting a behaviour style that better represents how they intend to face the decision-making process, select an expertise level about the topic being discussed and select which decision-makers involved in the same process they consider to be more credible. CAP is an analytic method that aims to provide a decision based on the values mentioned in the previous configurations and that maximizes the satisfaction level of the group and of each one of the decision-makers [71].

2.4.3.5 WSM (Weighted Sum Model)

Weighted sum model is the best known and simplest multi-criteria decision analysis method for evaluating a number of alternatives in terms of a number of decision criteria [72]. Supposing that a given MCDA problem is defined on m alternatives and n decision criteria and assuming that all the criteria are benefit criteria, that is, the higher the values are, the better it is, W_j denotes the relative weight of importance of the criterion C_j and a_{ij} is the performance value of alternative A_j when it is evaluated in terms of criterion C_j , the total (i.e., when all the criteria are considered simultaneously) importance of alternative A_i, denoted as $A_i^{WSM-score}$, is defined by the following equation:

$$A_i^{WSM-score} = \sum_{j=1}^{n} W_j a_{ij} \, , for \; i = 1,2,3, \dots, m \qquad (1)$$

For the maximization case, the best alternative is the one that yields the maximum total performance value and it is important to state that it is applicable only when all the data are expressed in exactly the same unit, meaning a normalization will be required. On the other hand, the non-beneficial attribute is one in which minimum values are desired and as this is:

1. For beneficial attributes, $X = x|xmax$
2. For non-beneficial attributes, $X = xmin|x$

2.5 Cased-Based Reasoning

Case-based reasoning (CBR) is a problem-solving paradigm that in many respects is fundamentally different from other major artificial intelligence approaches. Instead of relying solely on general knowledge of a problem domain or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case and reusing it in the new problem situation. A second important difference is that CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems. The CBR field has grown rapidly over the last few years, as seen by its increased share of papers at major conferences, available commercial tools, and successful applications in daily use [73]. Accordingly with Aamodt and Plaza, 1994 [73] and at the highest level of generality, a CBR cycle may be described by the following four processes:

- Retrieve: the most similar case or cases;
- Reuse: the information and knowledge in that case to solve the problem;
- Revise: the proposed solution;
- Retain: the parts of this experience likely to be useful for future problem solving.

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledgebase (case-base). The four processes each involve a number of more specific steps, which will be described in the task model. In Figure 1, this cycle is illustrated.

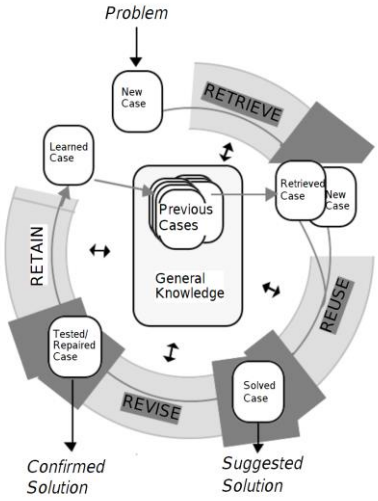

Figure 1 The CBR Cycle[1]

An initial description of a problem (top of figure) defines a new case. This new case is used to Retrieve a case from the collection of previous cases. The retrieved case is combined with the new case - through REUSE - into a solved case, i.e., a proposed solution to the initial problem. Through the REVISE process this solution is tested for success, e.g., by being applied to the real-world environment or evaluated by a teacher, and repaired if failed. During RETAIN, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases. As indicated in the figure, general knowledge usually plays a part in this cycle, by supporting the CBR processes. This support may range from very weak (or none) to very strong, depending on the type of CBR method. By general knowledge we here mean general domain-dependent knowledge, as opposed to specific knowledge embodied by cases. For example, in diagnosing a patient by retrieving and reusing the case of a previous patient, a model of anatomy together with causal relationships between pathological states may constitute the general knowledge used by a CBR system [73].

The implementation of CBR inside informatic systems require the implementation of algorithms to calculate the similarity between objects. For the purpose of this solution, three different methods based on mathematical formulas were study in order to determine similarity values.

2.5.1 Euclidean Distance

Euclidean Distance defines the distance between two points in the same space, through the measurement of the width of the line that connects those points [74].

[1] Adapted from [64].

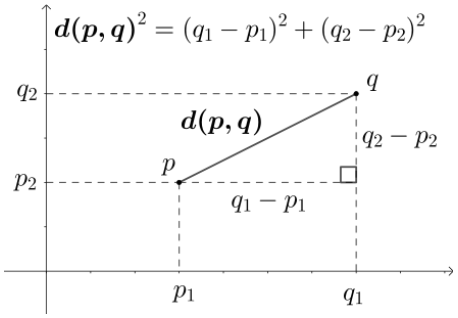

Figure 2 Euclidean Distance between two points[2]

For a space of n dimensions and P points, with following coordinates (P_1, P_2,…, P_n), and Q, with coordinates (Q_1, Q_2,…, Q_n), the distance is given by the following formula:

$$d\,(\mathrm{P},\mathrm{Q}) = \sqrt{(Q_1 - P_1)^2 + (Q_2 - P_2)^2 + \cdots + (Q_n - P_n)^2} = \sqrt{\sum_{i=1}^{n}(Q_i - P_i)^2}, \quad (2)$$

In the scope of data mining which is the process of extracting and discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems, Euclidean Distance is the technique used to measure similarity between two objects [75], where the information of the objects are defined through vectors and a formula is then applied in order to verify distance between those objects. The closer the distance value are from 0, the more similarity exists between two objects.

2.5.2 Cosine Similarity

It's a technique used to determine similarity between two objects, represented by vectors [76], based on the cosine angle formed by the two objects. For two vectors, $\vec{a}$ and $\vec{b}$, the similarity is given by the following formula:

$$\cos(\theta) = \frac{\vec{a}.\vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{a_1 b_1 + a_2 b_2 + \cdots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} + \sqrt{b_1^2 + b_2^2 + \cdots + b_n^2}} \quad (3)$$

The similarity value is within -1 and 1, where -1 is equivalent to the 180º cosine, meaning, the vectors have opposite directions, and where 1 is equivalent to the 0º cosine, meaning, the vectors have the same direction. The closer to 1 is the cosine angle formed by the two vectors, the grater is the similarity between those objects. In the following figure are illustrated the angles between two vectors, being vector *a* and *b* in opposite directions, and so not similar, or with same direction, meaning they are similar to each other.

[2] Adapter from [125]

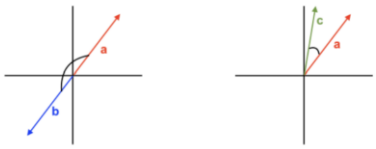

Figure 3 Example of angles between vectors[3]

This technique is one of the most popular ones in text analysis in documents, being applied in order to aggregate and classify documents [77]. This technique could be also integrated in recommender systems, for instance, to generate recommendations for movies based on preferences provided by other users that classified the movie in a similar way the user already classified [78].

2.5.3 Jaccard Index

The Jaccard similarity index (sometimes called the Jaccard similarity coefficient) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations. Although it's easy to interpret, it is extremely sensitive to small samples sizes and may give erroneous results, especially with very small samples or data sets with missing observations [79]. For sets based on binary values the following formula is used to determine similarity:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad (4)$$

Where:

- |A∩B| represents the number of equal elements in both sets;
- |A∩B| represents the total number of elements in both sets.

For the determination of similarity between two sets of real or positive values, x and y, we have the following formula [79]:

$$J(x,y) = \frac{\sum_i min(x_i, y_i)}{\sum_i max(x_i, x_i)} \qquad (5)$$

The result value obtained by this formula is between 0 and 1, where 0 indicates that sets don't share any member and 1 indicates that the sets are equal. In sum, the closest to 1 the grater is the similarity between sets.

2.5.4 K-Nearest Neighbors

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand but has a major drawback of becoming significantly slows as the size of that data in use grows [80].

[3] Adapted from [126]

2.6 Multi-Agent Systems

Multi-agent systems are systems composed of multiple interacting computing elements, known as agents. Agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action – of deciding for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents - not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation, and the like. Multi-agent systems are a relatively new sub-field of computer science, they have only been studied since about 1980, however, since then international interest in the field has grown enormously. This rapid growth has been spurred at least in part by the belief that agents are an appropriate software paradigm through which to exploit the possibilities presented by massive open distributed systems – such as the Internet. Although they will certainly have a pivotal role to play in exploiting the potential of the Internet, there is a lot more to multiagent systems than this. Multiagent systems seem to be a natural metaphor for understanding and building a wide range of what we might crudely call artificial social systems. The ideas of multiagent systems are not tied to a single application domain, but, like objects before them, seem to find currency in a host of different application domains [81].

2.6.1 Agents

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment to meet its design objectives [85]. Agents can interact with environments through functional systems or reactive systems. Agents are autonomous because they can operate without direct intervention by humans or others, they can interact with other agents or even less with humans, they are reactive and can collect information from the environment and react to it, and in addition to reacting to the environment, they have the initiative to exhibit goal-directed behavior.

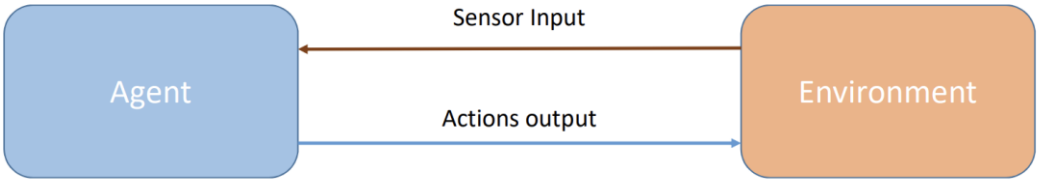

Figure 4 Agent understand environments and interact with them

2.6.2 Environment

The environments are usually characterized by the following properties:

- Accessible versus Inaccessible: An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state. Most real-world environments (including, for example, the everyday physical world and the Internet) are not accessible in this sense [85];

- Deterministic versus non-deterministic: A deterministic environment is one in which any action has a single guaranteed effect - there is no uncertainty about the state that will result from performing an action [85];

- Static versus dynamic: A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent. In contrast, a dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment, as is the Internet [85];

- Discrete versus Continuous: An environment is discrete if there are a fixed, finite number of actions and percepts in it [85].

2.6.3 Agents and Objects

Objects are defined as computational entities that encapsulate some state, are able to perform actions, or methods on this state, and communicate by message passing. The difference between agents and objects are [85]:

- Degree of autonomy;

- Exhibition autonomy over its state;

- An object does not exhibit control over its behavior.

2.6.4 Agents and Expert Systems

Expert systems do not interact directly with any environment, they get their information not via sensors, but through a user acting as middleman. Expert systems do not act on any environment, but rather give feedback or advice to third-party expert systems that are not generally equipped with social ability, in the sense of cooperation, coordination, and negotiation [85].

2.6.5 Agents and Distributed Systems

Modules in distributed computing applications do not constitute agents. They are rarely “smart”, and hence much less robust than agents are (or should be). In agent-based systems generally, the communication involves high-level messages in contrast to the low-level

messaging in distributed computing. The use of high-level messaging leads to lower communication costs, easy re-implementability and concurrency. Agent-based applications operate typically at the knowledge level, not at the symbol level as is the case in distributed computing applications. In any case, modules in distributed computing applications are not autonomous in the same sense as for agent applications [85].

2.6.6 Agents Communication

Following will be presented some of the more relevant protocols used for the interaction between agents, through the use of specific language for the message exchange, which allow agent to share information independently of its structure or platform used to represent knowledge.

2.6.6.1 KIF

KIF (Knowledge Interchange Format) is a protocol to exchange and share information between agents, proposed by Genesereth and Fikes in 1992 [82]. This protocol present declarative semantics by using expressions that can easily understood without being manipulated by the interpreter. The logic used in KIF is of first order, which allow to represent meta-knowledge. Some message types can be exchanged using this protocol, like:

- Notification – (tell (>100 10));
- Request – (perform (print “Hello World” t));
- Question – (ask-if(>100 10));
- Reply – (reply true);
- Subscription – (subscribe (coordinates ?a ?b ?c)).

2.6.6.2 KQML

KQML (Knowledge Query Manipulation Language) was developed for the project KSE (Knowledge Sharing Effort) and introduced in 1993 [83] and like KIF it also presents a declarative semantic.

KQML language is composed by three layers:

1. Content: represent an expression or reason of communication and it can be expressed in multiple languages like KIF or Prolog;
2. Message: defines the logic communication type, as well as, all types of interactions allowed with an agent that also communicate with KQML;
3. Communication: define how the communication will be stablished, including sender identification, recipient, communication type among other details.

This language is composed by a variety of primitive sets that could be:

1. Informative: tell, achieve, untell, unachieved;
2. Requests: perform;
3. Questions: ask-if, ask-one, ask-all, evaluate;
4. Replies: reply, sorry;
5. Capabilities: subscribe, monitor, advertise;
6. Communications: forward, register, broadcast.

The following example illustrates a dialog between two agents A and B, using KQML, where agent A asks agent B if 100 is superior to 10, and the positive reply by agent B:

- A to B: (ask-if(> 100 10)):
- B to A: (reply true).

2.6.6.3 ACL

ACL (Agent Communication Language) it's a communication language between agents and was introduced by FIPA [84]. The language is defined by three main components: a vocabulary, an internal language (KIF) and an external language (KQML). One ACL message corresponds to a KQML expression with arguments in the KIF format and is composed by words defined in its vocabulary.

Among other characteristics, ACL language should be able to:

1. Handle prepositions, rules and actions instead of objects, focusing only on the semantic and word meaning;
2. Describe desire states in a declarative language, instead of procedures or methodologies.

2.6.7 Agents Typology

Accordingly, with Nwana et al, 1996 [85], agents have individual internal states and goals, and they act in such a manner as to meet its goals on behalf of is user. They interact with other agents and possibly humans via some communication language and learn as they react and/or interact with their external environment.

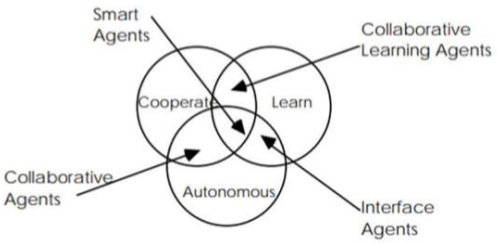

Figure 5 Agents Typology[4]

2.6.7.1 Collaborative agents

Collaborative agents are defined with emphasis on autonomy and cooperation. They may learn, but this aspect is not typically a major emphasis of their operation. Creating a system that interconnects separately developed collaborative agents, enable ensemble to function beyond the capabilities of any of its members.

Formally,

$$V(agent_i) > max\big(V(agent_i)\big) \qquad (6)$$

Where V represents "value addeness".

As motivations these agents are defined by:

- solve problems that are too large for a centralized single agent to do;
- allow interconnecting and interoperation of multiple existing legacy systems, e.g., expert systems, decision support systems, etc;
- provide solutions to inherently distributed problems, e.g., distributed sensor networks or air-traffic control;
- provide solutions which draw from distributed information sources, e.g., for distributed on-line information sources;
- provide solutions where the expertise is distributed, e.g., in health care provisioning;
- enhance modularity (which reduces complexity), speed (due to parallelism), reliability (due to redundancy), flexibility (i.e., new tasks are composed more easily from the more modular organization) and reusability at the knowledge level (hence shareability of resources);
- research into other issues, e.g., understanding interactions among human societies.

In order to have a coordinated set up of collaborative agents, they may have to negotiate to reach mutually acceptable agreements on some matters, and so they need to have strong

[4] Adapted from [85]

definitions, e.g., some attribute mentalistic notions such as beliefs, desires and intentions – yielding BDI-Type collaborative agents.

2.6.7.2 Interface agents

These agents are focused on autonomy and learning in order to perform tasks for their owners. They support and provide assistance, providing personal assistance to who is collaborating with the user in the same work environment. Collaborating with a user may not require an explicit agent communication language as one required when collaborating with other agents. Figure 6 adapted from [85] illustrated the interface agents.

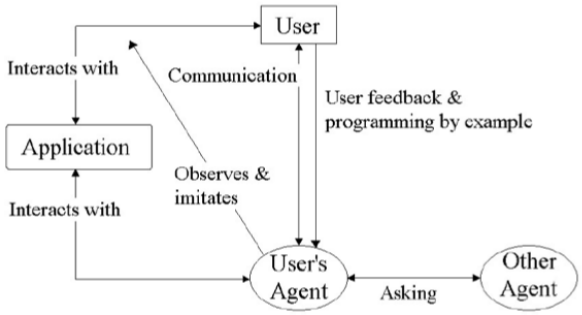

Figure 6 Interface agents[5]

These agents learn by observing and imitating the user, through receiving positive or negative feedback from the user. They also learn by receiving explicit instructions from the user and by asking other agent for advice. Cooperation with other agents, if any, is limited typically to asking for advice, and not in getting into protracted negotiation deals with them as is the case with collaborative agents.

2.6.7.3 Mobile agents

These agent type could be defined as computational software processes capable of roaming wide area networks such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner, and coming "back home" having performed the duties set by its user. Their mobility is neither a necessary nor sufficient condition for agenthood. Mobile agents are considered agents because they are autonomous and they cooperate, albeit differently to collaborative agents.

2.6.7.4 Information/Internet agents

These agents could manage the explosive growth of information, acting as role managers, manipulating, or collating information from many distributed sources. Interface or collaborative agents started out quite distinct, but with the explosion of the WWW and because of their applicability to this vast WAN, there is now a significant degree of overlap. These agents must be endowed with the capabilities of knowing where to look, how to find the information and how to collate it. They vary in characteristics as they may be static or mobile, non-cooperative or social and they may or may not learn.

[5] Adapted from [85]

2.6.7.5 Reactive agents

These agents are characterized by not possessing internal, symbolic models of their environments. They act/respond in a stimulus-response manner to the present state of the environment in which they are embedded. They are relatively simple, and they interact with other agents in basic ways, nevertheless, complex patterns of behavior emerge from these interactions when the ensemble of agents is viewed globally.

2.6.7.6 Hybrid agents

Hybrid agents are a combination of two or more agent philosophies within a singular agent. The key hypothesis for having hybrid agents or architectures is the belief that, for some application, the benefits accrued from having combination of philosophies within a singular agent is greater than the gains obtained from the same agent based entirely on a singular philosophy. For instance if consider an agent based on both the collaborative and reactive philosophies, the reactive component, which would take precedence over the deliberative one, brings about benefits like robustness, faster response times and adaptability, as the deliberative part of the agent would handle the longer term goal-oriented issues. This hybrid agents are typically within hybrid architectures and end up having a layered architecture.

2.6.8 Negotiation

One of the most important aspects in multi-agent systems is related with the autonomy and intelligence needed by the agents in order to make decisions in conjunction with other agents. Agents represent real decision makers and they should have the intelligence and capacity to defend the best interests of their decision makers. In order to do that, they need sometimes to negotiate. The definition set to the negotiation process between agents is not universal and different vision of the topic have been presented over the years. In 1999, Beer and their colleagues [86] identified the negotiation between agents as “a key form of interaction that enables groups of agents to arrive at a mutual agreement regarding some belief, goal or plan”. In 2001, Jennings and his colleagues [87] defined negotiation as being “a distributed search through a space of potential agreements”. Later in 2003, Rahwan and is colleagues [88] described negotiation as “a form of interaction in which a group of agents, with conflicting interests and a desire to cooperate, try to come to a mutually acceptable agreement on the division of scarce resources”. Back in 2011 Hadidi and his colleagues [89] defined negotiation as “the process of looking for an agreement between two or several agents on one or more issues”. Apart from the difference between all the definitions, some ideas are similar, which allow us to conclude that negotiation between agents it’s a process that include at least two agents, and it will be defined by the interactions between those agents in order to achieve mutual agreement. That agreement can include information and resource sharing and will be cases where both agents will be forced to sacrifice some of their interests so that a beneficial agreement could be done.

Beer and his colleagues [86] identified three main research areas related with negotiation, which are negotiation protocols, negotiation objects and reasoning models. As for the negotiation protocols they define a set of rules that govern the interactions between agents, including participant types, negotiation states and allowed actions accordingly with the participant type and the considered state. Sarit Kraus in 2001 [90] questioned the parameter types that should be used in negotiation protocols and identified seven parameters,

Distributions, symmetry, negotiation time, efficiency, simplicity, stability, and money transfer. The negotiation objects include the problems for which agents should agree, like price, quality, etc. At last reasoning models are decision making instruments used by the agents to achieve their objectives.

As follow will be presented some of the main approaches to automated negotiation in the multi-agent literature.

2.6.8.1 Game-theoretic approach

Game theory has its roots in the work of Neuman and Morgenstern (1944) [91] and it's a branch of economics, that studies the strategic interactions between self-interested economic agents. More recently it has been used extensively to study interactions between self-interested computational agents. According with Jennings and colleagues given the negotiation scenario that includes the use of automated agents, game-theory could be used to handle two types of problems:

1. Protocol definition that limits the proposals the agents can exchange during negotiation process and that include multiple mechanisms like "guaranteed success"," maximizing social welfare", "individual rationality", and others;

2. Strategy definition to adopted by individual agents during negotiation, maximizing their welfare. The issue with this process is related with the fact that sometimes the defined strategy even though seems ideal, is impossible to translate into reality within a computational environment.

The main problems associated with this negotiation type are [87]:

1. Game theory assumes that it is possible to characterize an agent's preferences with respect to possible outcomes, meaning that with more complex (multi-issue) preferences, it is much harder to use them;

2. The theory has failed to generate a general model governing rational choice in inter-dependent situations and instead the discipline has produced a number of highly specialized models that are applicable to specific types of interdependent decision making;

3. Game theory models often assume perfect computational rationality meaning that no computation is required to find mutually acceptable solutions within a feasible range of outcomes. This assumption is rarely true in most real world cases and agents typically know their own information space, but they do not know that of their opponent.

2.6.8.2 Heuristic approaches

Heuristic approaches emerged as a new type of negotiation that intents to overcome many of the game-theory limitations, mainly what concerns to computational costs and decision-making actions needed. In this approach the goal is not looking for optimal solutions, but rather solutions that are good enough taking in account lower reasoning agents capability and available resources [88].

Jennings and colleagues [87] mentioned the main advantages in the use of this type of negotiation as being:

1. The models are based on realistic assumptions; hence they provide a more suitable basis for automation, and they can, therefore, be used in a wider variety of application domains;

2. The designers of agents, who are not wedded to game theory, can use alternative, and less constrained, models of rationality to develop different agent architectures.

As mentioned by Jennings and colleagues [87] the space of possible agreements is quantitatively represented by contracts having different values for each issue. Each agent then rates these points in the space of possible outcomes according to some preference structure, captured by a utility function. Proposal and counter-proposal are then offers over single points in this space of possible outcomes, and search terminates either when the time to reach an agreement has been exceeded or when a mutually acceptable solution, a point of intersection of the agents' acceptable outcomes sets, has been reached.

Some limitations of this approach are:

1. The models often select outcomes (deals) that are sub-optimal and this happens because they adopt an approximate notion of rationality and because they do not examine the full space of possible outcomes;

2. The models need extensive evaluation, typically through simulations and empirical analysis, since it is usually impossible to predict precisely how the system and the constituent agents will behave in a wide variety of circumstances.

2.6.8.3 Argumentation-based approaches

In the previous techniques discussed the focused was on the trading of proposals, but they have three main limitations:

1. The proposals themselves generally denote single points of the space of negotiation agreements, meaning that agents are not allowed to exchange additional information;

2. The only feedback that can be made to a proposal is a counter-proposal, which itself is another point in the space, or an acceptance or withdrawal;

3. Its hard to change the set of issues under negotiation in the course of a negotiation, limiting their capacity of negotiation.

To remove this limitations, argumentation-based negotiation was created, allowing agents to exchange additional information along with the proposals sent. This enables agents to explain or justify their opinion in the form of arguments. In this way an agent that receives and rejects a proposal is able to justify why is not accepting it. This new way of exchange proposals during the negotiation process allow to redefine, not only the space of acceptance of each agent, through the redefinition of preferences), but also the negotiation space itself.

Kraus and colleagues in 1998 [92] identified six main types of arguments exchanged between agents using a language that represents beliefs, desires and intentions. This argument types are

similar to the ones used by people on a daily basis and are exchanged in a interactive way accordingly with their order of strength. The identified arguments are:

1. Appeal to “prevailing practice” - Used when an agent verifies that the opposing agent will refuse to act in a certain way because it contradicts their goals. So the agent appeals to the fact that in the past another agent has agreed to act in this way under the same conditions;

2. Counterexample - Used when an agent places an order to another agent and the request is refused because it contradicts its aims or intentions. The agent claims as a counterexample that in the past this agent acted in a way that also went against their goals and intentions;

3. Appeal to past promise - Used when an agent expects another agent to accept his request based on some promise made in the past;

4. Promise of a future reward - Used when an agent expects another agent to accept his request by promising to act in a certain way in the future that will contribute to that agent desires;

5. Appeal to self-interest - Used when an agent expects another agent to accept his request by appealing it would be in his best interest to accept such request;

6. Evaluation of threats – Used when an agent expects another agent to accept his request promising to act in a certain way in the future, which will contradict his desires.

By using argumentation models, agents are capable to defend the interests of those who they represent, and also justify and support their ideas and actions. However, regardless of how much knowledge they might hold, it is essential to define their behavior [93]. An adaptation of a conflict management model to the context of Group Decision Support Systems, can be approach by:

1. Reflect a natural way of human behavior in the agents;

2. Provides an easier way to reach an agreement between all parties involved;

3. Not have high configuration costs to the participants.

This approach will offer a simple yet perceptible configuration tool that can be used by the participants and contribute to more intelligent communications between agents and makes possible for the participants to have a better understanding of the types of interactions experienced by the agents belonging to the system [93]. The intend to define valid behaviors for agents in group decision-making context and to relate the theoretical behaviors definition with usual attitudes and acts that are relevant for this context can be defined by two dimensions and relate them with two facets based on the Five Factor Model. The behaviors classification can be defined in three different levels (low, moderate and high) for each one of the dimensions. In order to classify the behaviors in a scale, the value of the personality trait correspondent to each facet [94].

Negotiation models inspired by the communication logic used in social networks can also be used, being the main idea defined by two main types of communication:

1. Public communication (PC), in the form of public posts;

2. Private communication (PrC), in the form of private chat.

The visual idea of this communication form is much alike to the one used, for instance, in Facebook. The fact of considering the way of communication used in social networks a good approach to serve as inspiration for this work topic is related to two main factors: the agents communicate in a context similar to the one practiced by the decision-makers in face-to-face meetings and the environment and the agents communication/interaction is easily understood by the participants (decision-makers) [95].

2.7 Sentiment Analysis

Sentiment Analysis (SA) or Opinion Mining (OM) is the computational study of people's opinions, attitudes and emotions toward an entity. The entity can represent individuals, events or topics. These topics are most likely to be covered by reviews. The two expressions SA or OM are interchangeable. They express a mutual meaning. However, some researchers stated that OM and SA have slightly different notions [96]. Opinion Mining extracts and analyzes people's opinion about an entity while Sentiment Analysis identifies the sentiment expressed in a text then analyzes it. Therefore, the target of SA is to find opinions, identify the sentiments they express, and then classify their polarity as shown in Figure 7. Sentiment Analysis can be considered a classification process as illustrated in Figure 7. There are three main classification levels in SA: document-level, sentence-level, and aspect-level SA. Document-level SA aims to classify an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit (talking about one topic). Sentence-level SA aims to classify sentiment expressed in each sentence. The first step is to identify whether the sentence is subjective or objective. If the sentence is subjective, Sentence-level SA will determine whether the sentence expresses positive or negative opinions. Wilson et al. [97] have pointed out that sentiment expressions are not necessarily subjective in nature. However, there is no fundamental difference between document and sentence level classifications because sentences are just short documents [98]. Classifying text at the document level or at the sentence level does not provide the necessary detail needed opinions on all aspects of the entity which is needed in many applications, to obtain these details; we need to go to the aspect level. Aspect-level SA aims to classify the sentiment with respect to the specific aspects of entities. The first step is to identify the entities and their aspects. The opinion holders can give different opinions for different aspects of the same entity like this sentence ''The voice quality of this phone is not good, but the battery life is long''. This survey tackles the first two kinds of SA.

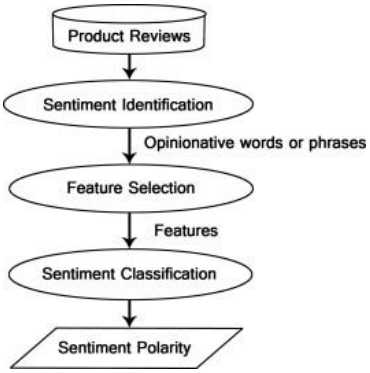

Figure 7 Sentiment analysis process on product reviews [6]

Sentiment Analysis task is considered a sentiment classification problem. The first step in the SC problem is to extract and select text features. Some of the current features are [99]:

- Terms presence and frequency: These features are individual words or word n-grams and their frequency counts. It either gives the words binary weighting (zero if the word appears, or one if otherwise) or uses term frequency weights to indicate the relative importance of features [100];

- Parts of speech (POS): finding adjectives, as they are important indicators of opinions;

- Opinion words and phrases: these are words commonly used to express opinions including good or bad, like or hate. On the other hand, some phrases express opinions without using opinion words. For example: cost me an arm and a leg;

- Negations: the appearance of negative words may change the opinion orientation like not good is equivalent to bad.

2.7.1 Feature selection methods

Feature Selection methods can be divided into lexicon-based methods that need human annotation, and statistical methods which are automatic methods that are more frequently used. Lexicon-based approaches usually begin with a small set of 'seed' words. Then they bootstrap this set through synonym detection or on-line resources to obtain a larger lexicon. This proved to have many difficulties as reported by Whitelaw et al. [101]. Statistical approaches, on the other hand, are fully automatic.

The feature selection techniques treat the documents either as group of words (Bag of Words (BOWs), or as a string which retains the sequence of words in the document. BOW is used more often because of its simplicity for the classification process. The most common feature selection step is the removal of stop-words and stemming (returning the word to its stem or root i.e., flies → fly).

[6] Adapted from [83]

2.7.2 Sentiment classification techniques

Sentiment Classification techniques can be roughly divided into machine learning approach, lexicon based approach and hybrid approach [102]. The Machine Learning Approach (ML) applies the famous ML algorithms and uses linguistic features. The Lexicon-based Approach relies on a sentiment lexicon, a collection of known and precompiled sentiment terms. It is divided into dictionary-based approach and corpus-based approach which use statistical or semantic methods to find sentiment polarity. The hybrid Approach combines both approaches and is very common with sentiment lexicons playing a key role in the majority of methods. The various approaches and the most popular algorithms of SC are illustrated in Figure 8.

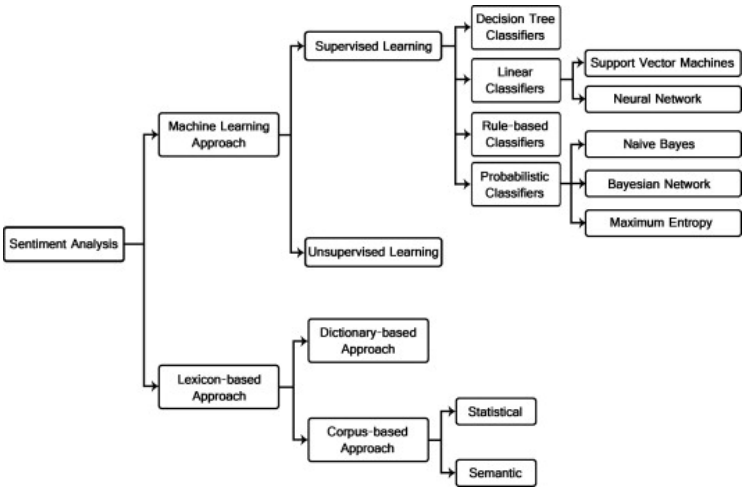

Figure 8 Sentiment classification techniques[7].

The text classification methods using ML approach can be roughly divided into supervised and unsupervised learning methods. The supervised methods make use of a large number of labelled training documents. The unsupervised methods are used when it is difficult to find these labelled training documents. The lexicon-based approach depends on finding the opinion lexicon which is used to analyse the text. There are two methods in this approach. The dictionary-based approach which depends on finding opinion seed words, and then searches the dictionary of their synonyms and antonyms. The corpus based approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus to help in finding opinion words with context specific orientations. This could be done by using statistical or semantic methods.

2.8 Related works

The literature review is structured on several sub-sections according to the relevant topics and methods used in this dissertation, such as different works regarding Multi-Criteria Decision-Making and Recommender Systems, aspects of doing Recommender Systems with the use of Multi-Agent Systems and some of its advantages and finally dives into why Sentiment Analysis could be an add-on on Recommender Systems, along with some existing methods and techniques that will be used during the implementation phase on chapter 4.

[7] Adapted from [87].

2.8.1 Multi-Criteria Decision-Making and Recommender Systems

As proposed solution, the diabetic-friendly system should be able to incorporate user preference information upon multiple criteria to best recommend. The system should predict ratings that affect the overall user profile preference values. In this sense approaching the problem with MCDM methods and techniques to implement the proposed system should be the one that provides best change of positive results.

In **Multi-Criteria Recommender Systems**, Adomavicius, Gediminas et al [103] article is provided an overview of the class of multi-criteria recommender systems. First, is defined the recommendation problem as a multi-criteria decision making (MCDM) problem, and reviewed MCDM methods and techniques that can support the implementation of multi-criteria recommender system. Then, the article focuses on the category of *multi-criteria rating recommenders*– techniques that provide recommendations by modelling a user's utility for an item as a vector of ratings along several criteria. A review of current algorithms that use multi-criteria ratings for calculating predictions and generating recommendations is provided.

Bernard Roy (one of the1960s pioneers in MCDM methods) includes four steps when analysing a decision-making problem [104]:

1. *Defining the object of decision*. That is, defining the set of alternatives (items) upon which the decision has to be made and the rationale of the recommendation decision.

2. *Defining a consistent family of criteria*. That is, identifying and specifying a set of functions that declare the preferences of the decision maker (targeted user) upon the various alternatives. These should cover all the parameters affecting the recommendation decision and be exhaustive and non-redundant.

3. *Developing a global preference model*. That is, defining the function that synthesizes the partial preferences upon each criterion into a model that specifies the total preference of a decision maker regarding a candidate alternative.

4. *Selection of the decision support process*. This covers the design and development of the procedure, methods, or software systems that will support a decision maker when taking a decision about the set of alternatives (items), in accordance to the results of the previous steps.

The article also refers that most of the recommender systems can be classified in the following three general categories:

- *Multi-attribute content preference modelling*. Even though these systems typically use single-criterion ratings (e.g., numeric or binary ratings), for any given user these systems attempt to understand and model the commonalities of multi-attribute content among the items the user preferred in the past, and recommend to the user the items that best match this preferred content. For example, in a movie recommender system, these commonalities may be represented by specific genres, actors, directors, etc. that the user's preferred movies have in common.

- *Multi-attribute content search and filtering*. These systems allow a user to specify her general preferences on content-based attributes across all items, through searching or filtering

processes (e.g., searching for only “comedy” movies or specifying that “comedy” movies are preferable to “action” movies), and recommend to the user the items that are the most similar to her preferences and satisfy specified search and/or filtering conditions.

- *Multi-criteria rating-based preference elicitation*. These systems allow a user to specify her individual preferences by rating each item on multiple criteria (e.g., rating the story of movie Wanted as 2 and the visual effects of the same movie as 5), and recommend to the user the items that can best reflect the user’s individual preferences based on the multi-criteria ratings provided by this and other users.

In most of the Recommender Systems is applied a two-phase algorithm process and include a prediction phase and a recommendation phase. The first phase calculates the predictions of ratings for the unknown items and the later the phase where the predictions are used to support the recommendations, for instance the user gets recommended a set of top-N items that maximizes his/her utility function (i.e., N items with highest-predicted ratings). The article also proposes algorithms and techniques to apply in these two phases.

2.8.2 Multi-Agent Systems and Recommender Systems

Multi-Agent Systems are already being applied in the development of Recommender Systems and they seem to be a perfect fit to deal also with Multi-Criteria Decision-Making problems. Next are presented some literature works that incorporate MAS and RS with great similarities with the proposed solution in this dissertation. The main advantages of using MAS is the capacity of having dedicated agents to specific tasks, like collecting and maintaining user profile preference information, or collecting sentiment analysis rankings from the identified platforms (Zomato, TripAdvisor), along with specific agents dedicated to the multi-criteria decision problem. Multi-Agent Systems are also characterized by decentralized data, computation is asynchronous, there is no global control of the system, and each agent has a limited point of view. These systems are suitable for solving problems with multiple solving methods, multiple perspectives and/or multiple entities.

Integrating a multi-agent recommendation system into a Mobile Learning Management System. Alfio Andronico et al. [105] presented a project of three Italian universities that integrates a multi-agent recommendation system that suggests interesting educational resources to students. This project attempts to interconnect m-learning technologies with e-learning, and e-learning is integrated in the information systems of academic institutions. This project used effective models for mobile learning, some learning processes in mobile learning environments, and integrated a multi-agent recommendation system into the mobile learning management system that suggests very useful information to the users of the system. They integrated InLinx, a multi-agent Web-based hybrid recommender system that provides an online bookmarking service in their m-learning architecture. They extended the Learning Management System (LMS) to mobile technologies to allow the users to interact with the systems using some mobile devices like PDAs, cellular phones, and others. Then, they integrated a multi-agent recommendation system, which collect data about the user’s behaviour, preferences and then suggests educational resources.

A Multi-Agent Recommender System. Eugénio Oliveira et al. [106] implemented a recommender system using autonomous agents to solve the problem of web adaptation. They presented a multi-agent approach using agents with two different algorithms (associative rules

and collaborative filtering) based on binary data that produces item-based recommendations and makes bids to provide the next set of recommendations to the user. Agents cooperate to base their bids on client's satisfaction instead of their own revenue and they shared the same data, but the results are not combined to provide recommendation. For evaluation of performance, they used precision and recall measures. Results show that this multi-agent approach combining different algorithms can improve user's satisfaction.

An Intelligent Multi-Agent Recommender System for Human Capacity Building. Vukosi N. Marivate et al. [107] proposed a multi-agent approach to recommend training courses to electrical and mechanical engineering professionals by using information customization as well as recommendations. The system has two main agents: the recommendation agent and the information retrieval agent. The first one proposes a personalized list of training modules and the second one searches a predefined list of services provider's websites for course information and updates. They used several programming languages in the implementation, but Ruby was the best for information retrieval and C++ for the search engine. The system is scalable and adaptable, so it can be adapted to many problem fields such as job searches, academic advising, business support systems, and others.

2.8.3 Sentiment Analysis

It has been proven to be a valuable technique the use of Sentiment Analysis for recommender systems. The main goal of a recommender system is to predict the preference for an item of a target user. In this dissertation scenario, the goal is to predict the best diabetic-friendly recommendations having in account both user preferences and restaurant reviews.

In many social networking services or e-commerce websites, users can provide text reviews, comments or feedback to the items. These user-generated texts provide a rich source of user's sentiment opinions about numerous products and items. Potentially, for an item, such text can reveal both the related feature/aspects of the item and the users' sentiments on each feature [108]. The item's feature/aspects described in the text play the same role with the meta-data in content-based filtering, but the former is more valuable for the recommender system. Since these features are broadly mentioned by users in their reviews, they can be seen as the most crucial features that can significantly influence the user's experience on the item, while the meta-data of the item (usually provided by the producers instead of consumers) may ignore features that are concerned by the users. For different items with common features, a user may give different sentiments. Also, a feature of the same item may receive different sentiments from different users. Users' sentiments on the features can be regarded as a multi-dimensional rating score, reflecting their preference on the items. Also, some challenges are faced when it comes to applying sentiment analysis on review, like spam and biased reviews and should be further researched in order to apply the best approach for this case scenario. Cases like long and short forms of user-generated text should be treated differently and some interesting results show that short-form reviews are sometimes more helpful than long-form [109] because it is easier to filter out the noise in a short form text.

In the following two literature references are presented works with different approaches and techniques, that should be included in the comparison work the author is doing for the proposed solution.

Sentiment analysis of customer reviews in zomato bangalore restaurants using random forest classifier, Jonathan, Bern et al [110] applies a set of methods as term frequency-inverse document frequency (TF-IDF) and random forest classifiers to perform sentiment analysis on Zomato reviews, allowing to understand positive, negative and neutral sentiment restaurant reviews. The study perform a discussion based on accuracy results from the machine model identifying potential improvements on results by using imbalanced dataset algorithms.

Twitter sentiment analysis of app based online food delivery companies, Trivedi, Shrawan Kumar et al [111] presents a study performed on three app-based food delivery companies, Swiggy, Zomato and UberEats. In this study Twitter was used as the data collection platform where customer's tweets related to all three companies are fetched. Lexicon-based sentiment analysis method is applied on the tweets fetched for the companies. A descriptive analytical method is used to compute the score of different sentiments. Lexicon-based sentiment classification is always preferable than machine learning or other model because it gives flexibility to make your own sentiment dictionary to classify emotions. The results of this study reveal the value of social media competitive analysis and show the power of text mining and sentiment analysis in extracting business value and competitive advantage.

All this literature references allowed to mature the initial thoughts on techniques and best approaches. The work of Adomavicius, Gediminas et al [88] was a clear indicator of the importance of applying multi-agent system to solve multi-criteria decision-making problems. In their scenario the category of multi-criteria rating recommenders was the selected method, and since the current proposed solution will try to handle similar scenarios, this category was also chosen for the initial approach implementation, in opposition to the other two categories, since multi-criteria rating was the best fit. In the work presented by Alfio Andronico et al. [90] its clear that multi-agent systems can be also a good fit for recommender systems, even if they are applied to mobile applications, what is one of the suggestions provided in this dissertation. The proposed work was implemented with the help of a web-based portal for the user interaction but can easily be changed to be integrated within a mobile application with the same rate of success. The works of Eugénio Oliveira et al. [91] and Vukosi N. Marivate et al. [92] two main leads were suggested. The first is related with the application of a collaborative filtering approach for the agents in the system and the second is the fact that different agents can have different internal objectives, but can work together in order to achieve a main goal, which in this case will be the approach selected for the MCDM agents defined for the negotiation process, where multiple instances of this agents, with different configurations (internal goals) will work to provide the user the best restaurant recommendations.

In relation to sentiment analysis the work of Jonathan, Bern et al [95] shows the application of term frequency-inverse document frequency (TF-IDF) method in application of sentiment analysis to reviews and in the work of Shrawan Kumar et al [96] is stated the advantage of text data mining and sentiment analysis in extracting business value and competitive advantage, two points also considered in the proposed solution, that implement a specific service agent (Reviews Agent), adding value to the mult-criteria decision making problem.

3 Solution Modeling

In this chapter will be presented the proposed solution for the present work. The chapter begins by identifying the main actors in the solution, analyze functional and non-functional requirements. Is then presented the solution architecture overview, along with the domain model and technologies that supported the solution implementation.

3.1 Requirements Analysis

For this section all the main actors within the proposed solution are identified, along with an analysis for the functional and non-functional requirements

3.1.1 Main Actors

For the proposed solution the following main actors are present:

- User – this entity represents a diabetic patient, or any person willing to receive recommendations for healthier restaurants based on their personal preferences;
- User Interface – it's a web-based platform that allow interaction between the user and the system. This user interface could be implemented within a mobile application or another type of implementations, like a smartwatch;
- Communication Interface – this represents the layer of communication between the web portal where the user interacts and the recommender system itself. This is a REST/API layer that exchange JSON messages;
- Recommendation System – in charge for all knowledge process, meaning, all the information generated by the user, all the internal agents and the recommendation management.

3.1.2 Functional Requirements

In this section we will define use cases for the user and for the recommendation system. The following diagrams illustrate each use case. In Figure 9 is presented the use case of interaction with the user interface module of the solution.

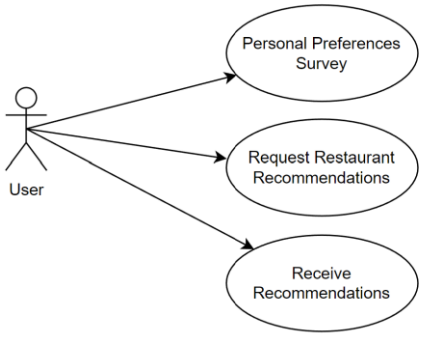

Figure 9 Use Case diagram for the user interacting with the user interface

In Figure 10 is represented the use case of interactions between the user interface and the recommendation system.

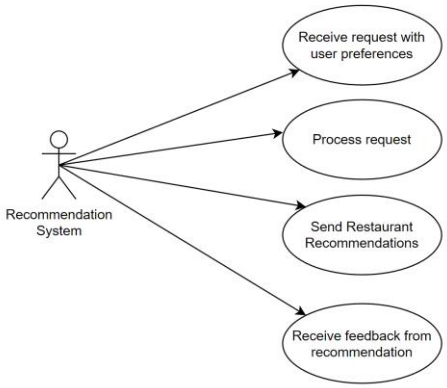

Figure 10 Use case diagram for the recommendation system

In the following sections are described the use cases from success scenarios and their respective system sequence diagram.

3.1.2.1 User Use Case

This use case includes all the process that occurs on the user side, in our case, a diabetic person, with the system. All interactions are currently done through a web portal but can be presented in many different ways. As it stands, its presented below a typical scenario of interaction:

- The user can start the system simulation by typing his/her name in the Nickname textbox presented. That nickname will act as an identifier and there cannot be two simultaneous users with the same nickname in the system. Next, the user will have a small survey composed by four questions answered by sliding the response control between one of two possible values. Those questions have the purpose to help the user to identify his/her own preferences. After the initial setup the user can start requesting restaurant recommendations by clicking the start button. In a while the system will start all the internal negotiation process in order to retrieve recommendations by one of the three defined categories (Best Choice, Nearby or Reviews).

Specified the use case scenario behind, the system sequence diagram representation is shown in Figure 11

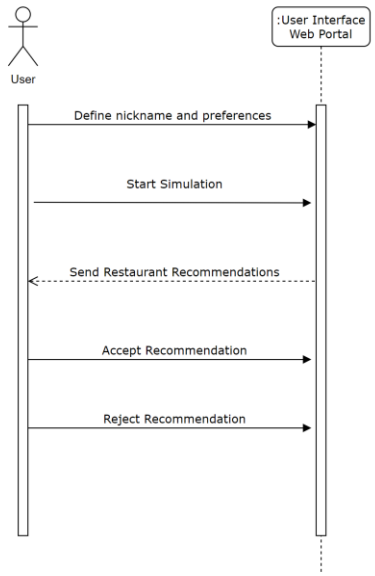

Figure 11 Use Case sequence diagram

3.1.2.2 Recommendation System Use Case

The second use case represents the interactions that occur between the web-portal and the recommender system module. All the communications use a REST/API layer that is bidirectional. Figure 12 shows the sequence diagram representation.

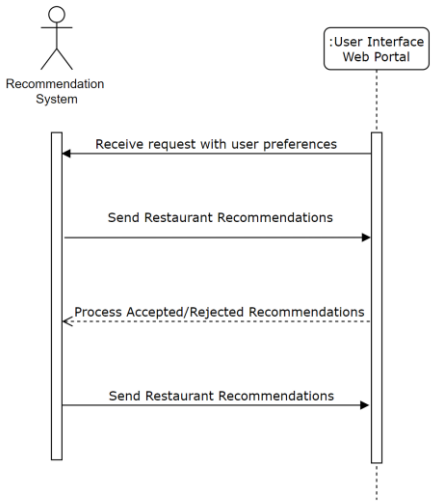

Figure 12 Recommendation System sequence diagram

3.1.3 Non-Functional Requirements

In this section will be identified the non-functional requirements for the proposed solution. Those requirements will be in line with quality attributes like:

1. Security;
2. Integrity;
3. Usability;
4. Reliability;
5. Performance;
6. Interface requirements.

The solution was developed in compliance with European RGPD rules and even though a minimal user data is collected, all the solution is RGPD compliant. The solution is designed not to record any data and all artificial intelligence mechanisms will run in real-time, through a distributed multi-agent system and web service communications. This contributes for the security and integrity of the information. On the usability aspect was identified that a simple and friendly web portal, easy to use would be best. As reliability was identified as a requirement to develop the system in consideration for the need of interoperability, meaning, the capability run the solution in different environments and hardware, creating a distributed solution, that can be a cloud solution as well as running on a local environment. That is also one of the gains of the solution when we look at the performance level. The multi-agent system base allows the system to communicate between different devices. Relative to interface requirements was identified as a channel for HTTP/REST communications to be enable between the web portal and recommendations system.

3.2 Solution Architecture

In this subsection is exposed the solution architecture for the solution proposed through a high-level diagram of components. They are briefly described as an introduction for the Chapter 4, where they will be detailed.

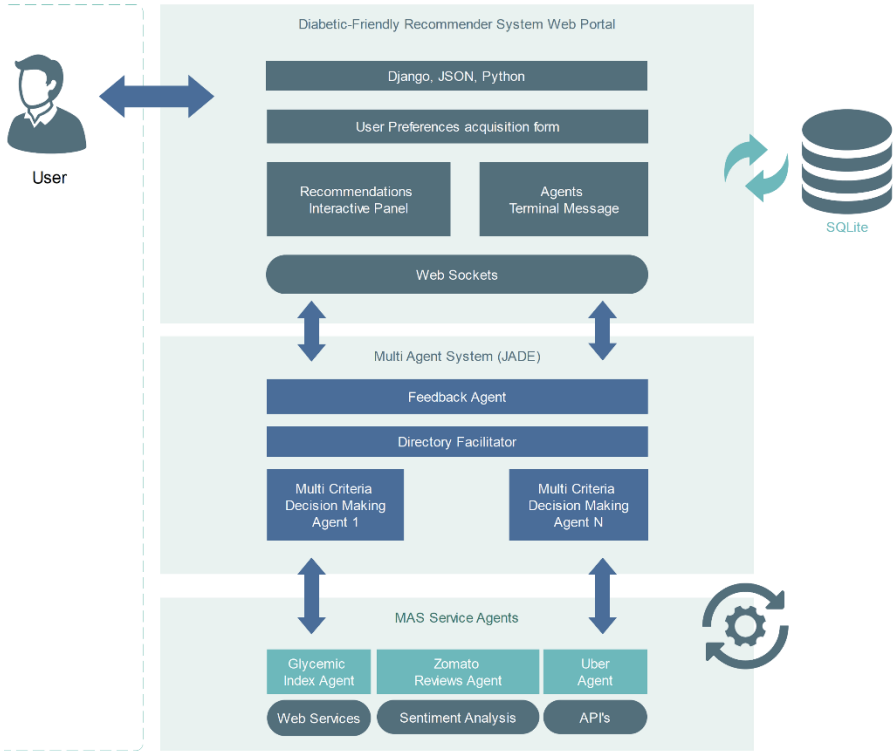

Figure 13 Solution Architecture

The solution proposed is split in three main components as demonstrated in Figure 13. The top component represents the User Interface Module, where all interactions between the user and the system occurs. The middle and bottom components shown in the figure, represent the Recommender System Module, that includes all the multi-agent system implementation.

3.3 Domain Model

In this subsection is presented an overview diagram of the main modules built for the solution.

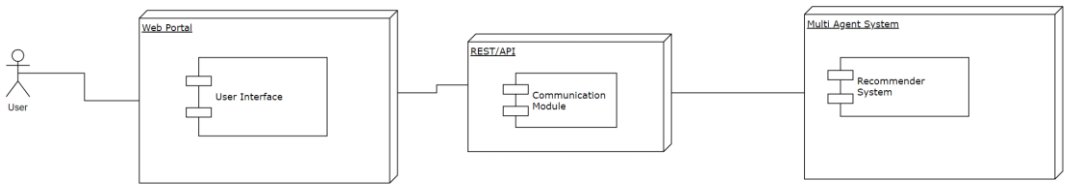

Figure 14 Solution Modules Diagram

In the figure above its possible to identify three main modules.

- User Interface Module: contains the web portal with which the user will interact;
- Communication Module: contains all the endpoints available for communications between the User Interface module and the Recommender System Module;
- Recommender System Module: represents the core module for the proposed solution, which includes the multi-agent system that provide restaurant recommendation to the user interface.

After this initial overview, it is also important to identify all the class representation created within the recommender system module and how they interconnect each other.

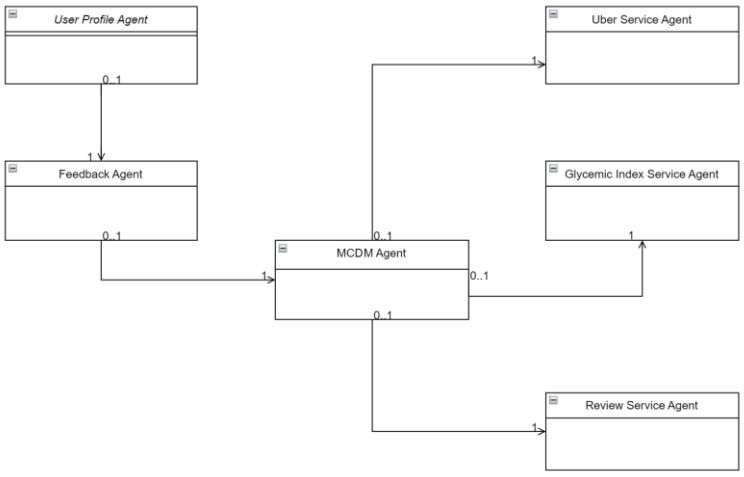

Figure 15 Class Domain Model for the solution

User Profile Agent will represent one user, containing all user preference information along with all previous interactions with the system.

Feedback User Agent represent a broker, it will be responsible to request MCDM Agents for restaurant recommendations for three different categories. While receiving restaurant

recommendations this agent will also use cased-based reasoning to find similarities between those received recommendations and previous history user interactions, allowing the system to evolve.

MCMD Agent represents one instance of the core of the negotiation process. These multi-criteria multi-decision-making agents have internal timers that define the maximum waiting time until proposals can be received before delivering them to the Feedback Agent. These agents can also be setup with specific goals, allowing different profiles of the same base MCDM Agent can exist simultaneously in the system improving the negotiation process.

Uber Service Agent is the entity that encapsulate two values to the negotiation process, price and distance. This entity provides restaurants recommendations with lower price and within a specific range, suggesting the user to get a meal by walking over, instead of ordering it, improving physical activity.

Glycemic Index Service Agent represents the service that provides the system with restaurant recommendations that depending on their menu and glycemic index for identified foods, will suggest better restaurant.

Reviews Service Agent represents the service that provides restaurant recommendations based on the sentiment analysis of their reviews.

3.4 Methods and Tools

In this subsection are identified the main technologies and tools involved in the development of the proposed solution.

3.4.1 JADE

To implement the base of the framework, was selected the use of JADE tool. JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middleware that complies with the FIPA specifications and through a set of graphical tools that support the debugging and deployment phases. A JADE-based system can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another, as and when required. JADE is completely implemented in Java language and the minimal system requirement is the version 5 of JAVA (the run time environment or the JDK) [112].

3.4.1.1 Agent Communication Language (ACL message)

As referenced in sub-section 2.6.6 there are already protocols for communication between the agents. For the purpose of the proposed solution, the Agent Communication Language was used. When one agent wants to send a message to another agent located on different container, a new ACL message must be created. An ACL message is a FIPA standard that defines different parameters used in agent communication. Only the performative and the receiver address parameter must be set. The remaining parameters are optional. The data being sent is

then coded with the proper codec and placed in the content slot of the message. The entire ACL message is then encoded.

3.4.1.2 Yellow Pages

The JADE platform implements the page service yellow on an agent: the Directory Facilitator (DF) agent, following the specifications of the FIPA standard. Agents who wish to publicize their services register with the DF, and the others can then search for agents who provide some desired service. In the multi-agent context, the requesting agent searches the platform's yellow pages by agents who offer a particular service. These agents must, when starting, register your services in the yellow pages. To graphically visualize this trade, we use a JADE platform agent that provides a graphical platform administration interface. It is called the Remote Management Agent (RMA) agent.

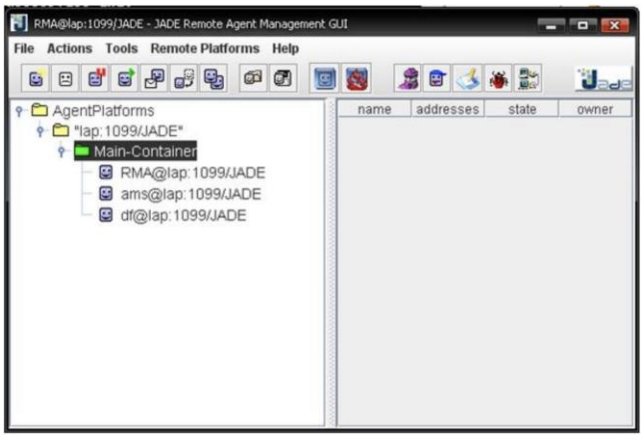

Figure 16 Remote Management Agent GUI

3.4.2 Visual Analogue Scale

The visual analogue scale (VAS) is a psychometric response scale that can be used in questionnaires. It is a measurement instrument for subjective characteristics or attitudes that cannot be directly measured. When responding to a VAS item, respondents specify their level of agreement to a statement by indicating a position along a continuous line between two end points. This was the select scale to be used to create the initial survey that users will fill-in to collect initial preferences.

3.4.3 IntelliJ IDEA

IntelliJ IDEA is an integrated development environment (IDE) written in Java. The selection of this IDE is directly connected with the fact that all the multi-agent system, that used JADE framework, is also built in JAVA. JAVA is an oriented-object programming language.

3.4.4 Django

As identified by the project itself, Django is a high-level python web framework for perfectionists with deadlines [113], which was found appropriate for the creation of the web portal module. Since this module could be replaced by any other form of user interface, the goal for this part of the project was to build a quick interface to interact with the recommender system.

3.4.5 WebSockets

WebSocket is a computer communication protocol, that provides bidirectional communication. The major advantage of it, is that enables interactions between a web browser and a web server, fitting the needs to implement the purposed solution, allowing real-time data transfer from and to the server.

3.4.6 Hibernate ORM and SQLite

Hibernate ORM enables developers to more easily write applications whose data outlives the application process. As an Object/Relational Mapping (ORM) framework, Hibernate is concerned with data persistence as it applies to relational databases (via JDBC) [114]. SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day [115]. Both tools were important in the development of the proposed solution as explained in more detail in Chapter 4.

3.4.7 jCOLIBRI

jCOLIBRI is a framework specifically designed for the development of Case-Based Reasoning Systems [116]. This tool was help application designer to develop and quickly prototyping CBR systems. jCOLIBRI has been designed as a wide spectrum framework able to support several types of CBR systems from the simple nearest-neighbor (*k-NN*) approaches based on flat or simple structures to more complex Knowledge Intensive ones. Other features of the framework like ontology integration, visualization of case bases, evaluation of CBR application, classification and maintenance methods are also included.

4 Solution Implementation

In this chapter is described all the development process for the proposed solution present in this dissertation, including justifications and decisions made during the implementation.

This chapter is divided in two main subsections, being the first one related with the process of implementation for the Recommender System Module (subsection 4.1) and the second related with the implementation of the User Interface Module (subsection 4.1.6). This organization was purposely done, so that the reader can understand deeper the more complex part of the solution and then the module the user will have to interact with the system. That also gives the needed line of thought to understand the chapter 5 where are present real simulation scenarios.

4.1 Recommender System Module

The recommender system module (RS Module) will describe the core module of the proposed solution. In this subsection will be detailed the implementation process, starting by defining each agent present in the system and they basic functions. Next will be explained why ACL messages were chosen for the protocol communication between agents, followed by a detail explanation on the negotiation process, the communication layer REST/API created to connect with the User Interface Module and last will be explained in more detail the sentiment analysis feature included in one of the service agents.

The core of the recommender system is a multi-agent system that contains a total of seven agents, being the last four consider service agents:

- User Profile Agent;
- Feedback User Agent;
- Multi-Criteria Decision Making Agent (MCDM Agent);
- Glycemic Index Feedback Agent;
- Uber Price Agent;
- Uber Distance Agent;
- Reviews Agent;

Each of these agents and their specific characteristics are next defined.

4.1.1 User Profile Agent

This agent represents the user, in this case a diabetic patient. The agent will be responsible to record the initial user preferences information, gather through the initial survey. This initial setup will allow the system to perform a personality analysis on the user, defining their base criteria, like if the user prefer restaurants closer by his location or if the user values opinion reviews about restaurants. This agent will only interact with the Feedback User agent. The recommendations for the user will improve on each interaction of the user with the system. It also works as a bridge between the multi-agent system and the user interface module through a set of web service endpoints.

4.1.2 Feedback User Agent

This agent can be viewed as the user representative or broker in the negotiation process. It will be responsible to negotiate which recommendations are the best for a particular user according with the user preferences best interest. During the negotiation process the agent will be capable of accept or refuse suggestions given by the multi-criteria decision-making agents. In this evaluation phase the agent will use a case-based reasoning algorithm to identify the best recommendation for the user based on his cases history log. After all this process this agent will provide the User Profile agent with restaurant recommendations for the three identified categories (Best Choice, Nearby and Reviews), and collect the user feedback for the provided recommendations. That information will update the user cases history log allowing the system to evolve over time.

4.1.3 Multi-Criteria Decision-Making Agent

This agent type is the core agent in decision-making process. It will have a specific profile based on the criteria weights, allowing the system to have multiple of these agents each one with different personal objectives, turning the negotiation process more diverse and complex. The MCDM agents will receive requests from the Feedback agent in representation of the user and each of this MCDM agents will request their preferred service agents for recommendations. Once they receive responses from the service agents, they start negotiating with the Feedback agent. During the negotiation process this agent will use argument-based negotiation providing the capability of explaining each recommendation gave and why it should be considering the best option for the user.

4.1.4 Service Agents

Service agents are responsible to provide MCDM agents the best restaurant option, based on the current user location and the criteria they are specialized in. In this sense we have four service agents, each one representing each criterion in consideration for the proposed solution as mapped in Table 1.

Table 1 Solution criteria for each recommendation category

Recommendation Category	Reviews	Glycemic Index	Price	Distance
Best Choice		Maximize	Minimize	
Nearby		Maximize		Maximize
Review	Maximize	Maximize		

Those four criteria will have maximization and minimization utility functions and are defined as:

- Reviews: will maximize the score value reviews of restaurants to provide the best resulted one. This agent holds internally the sentiment analysis feature to return the score value;
- Glycemic Index: will select restaurants where the foods with lower GI will be preferred for the recommendation;
- Price: will minimize the price cost of meals for the user;
- Distance: will optimize the distance and provide recommendations on restaurants that user can walk to, instead of ordering the meal, so in a sense will maximize distance, but in a defined range.

4.1.4.1 Reviews Agent

This agent is specialized in restaurant reviews. It will receive request from the MCDM agents requesting the best restaurant based on their reviews and in line with the user preferences. The agent has an internal sentiment analysis function that will use the ten closest restaurants and the latest ten reviews for each restaurant to retrieve the best scored one. This result will then be sent to MCDM agents as valid recommendation options.

4.1.4.2 Glycemic Index Feedback Agent

This is the specialized agent for the criteria Glycemic Index. As the other service agents, this one will take in consideration user preferences and present restaurant recommendation that sell meals with lower glycemic index values, assuming those will provide healthier diets for the user.

4.1.4.3 Uber Price Agent

This agent, specialized in the price criteria, will provide restaurant recommendation where the price is minimized for the best interest of the user, since this criterion is not related with health it should provide the system indicators of analysis that could influence healthier user decision.

4.1.4.4 Uber Distance Agent

The purpose of this agent, specialized in the distance criteria, is to maximize the distance at which the user is from the recommended restaurant, but in a walkable range. The idea behind

is to recommend restaurants the user can walk to, instead of ordering a meal, providing a physical activity to the user that produce health benefits.

4.1.5 Model Classes

In this subsection are explained the classes that define each of the agents present in the solution, as well as their main methods.

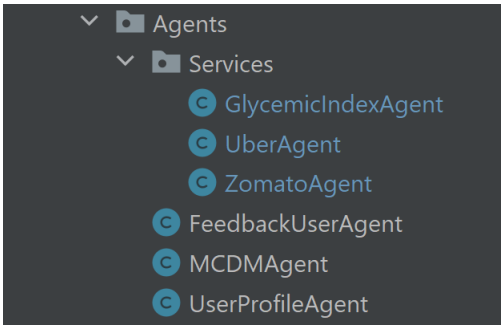

Figure 17 Agents classes

All the agents present in the solution derive from the **BaseAgent** class. This class extends the original JADE core agent implementation with a set of new functionalities added purposely for this framework solution. The BaseAgent class holds methods that are common to all the agents, like the ability to register services in the Directory Facilitator (Yellow Pages), as well as a method to search for a specific service on the Directory Facilitator (DF).

```
public class BaseAgent extends Agent{

    /** Method to Show basic Agent Detail Information */
    Bruno Teixeira (ISEP)
    public void ShowAgentDetails()
    {...}


    /** Method to register service on Directory Facilitator */
    5 usages  Bruno Teixeira (ISEP)
    public void RegisterServices(ArrayList<MessageProtocol> agentServices) {...}

    /** Method to search a service on Directory Facilitator */
    13 usages  Bruno Teixeira (ISEP)
    public void SearchForService(ServiceDescription service, MessageProtocol request) {...}

    /** Method to handle Agent termination */
    Bruno Teixeira (ISEP)
    protected void takeDown ( ) {...}

}
```

Figure 18 BaseAgent class

As next, are defined the common methods across agents present in the system. Those methods allow agent initializations and multiple handlers to treat each ACL message exchanged in the communication process. In the Figure 19 is shown the initialization for Reviews Agent.

```java
protected void setup() {
    Object[] args = getArguments();
    myZomato = (Zomato) args[0];

    try {

        RegisterServices(myZomato.getServicesProvided());
        HandleMessageReception();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 19 Example of agent initialization process

The initialization includes the registration of the agent in the system and the method for handling Agent Communication Language (ACL) messages.

```java
private void HandleMessageReception() {

    Bruno Teixeira (ISEP)
    addBehaviour(new CyclicBehaviour( a: this) {
        Bruno Teixeira (ISEP)
        public void action() {

            ACLMessage msg = receive();
            if (msg != null) {
                if (msg.getPerformative() == ACLMessage.INFORM) {
                    Handle_INFORM_MSG(msg);
                }else if (msg.getPerformative() == ACLMessage.REQUEST) {
                    Handle_REQUEST_MSG(msg);
                } else if (msg.getPerformative() == ACLMessage.PROPOSE) {
                    Handle_PROPOSE_MSG(msg);
                } else if (msg.getPerformative() == ACLMessage.ACCEPT_PROPOSAL) {
                    Handle_ACCEPT_PROPOSAL_MSG(msg);
                } else if (msg.getPerformative() == ACLMessage.REJECT_PROPOSAL) {
                    Handle_REJECT_PROPOSAL_MSG(msg);
                }
            } else
                block();
        }
    });

}
```

Figure 20 Message handler method present on all agents

4.1.6 ACL Messages

Agent Communication Language was used as the protocol for communication between the agent in the system. As base performative message types the following were used:

- INFORM: this performative was used every time an agent request to the Directory Facilitator (DF) for a specific service. The DF will trigger this performative message to all the agents in the system that provide the requested service;

- REQUEST: this performative was used after the identification of the agents that provide the requested service. This is the performative used in the first direct communication between agents, without the DF intervention;

- PROPOSE: this performative identifies messages that contain proposals, for restaurant recommendations;

- ACCEPT_PROPOSAL and REJECT_PROPOSAL: are the performatives that service agents receive from the MCDM agents, refusing or accepting the provided restaurant recommendations.

Apart from the performative message types, an enumeration was created to identify specific message, that are specific of the proposed solution. The **MessageProtocol** enumeration include the following message types:

- Recommendations: message protocol used by the User Agent to request restaurant recommendations to the Feedback User Agent;

- Best_Choice: message protocol to identify requests related with restaurant recommendations for the category “Best Choice”;

- NearBy: message protocol to identify requests for Nearby restaurant recommendations;

- Review: message protocol to identify requests for Reviews category;

- Zomato_RestaurantReviewScore: message protocol for messages used between the service agent Reviews and the MCDM agents;

- Glycemic_Index_Score: message protocol for messages used between the service agent Glycemic Index and the MCDM agents;

- Uber_BestPrice: message protocol used between the service agent Uber Price and the MCDM agents;

- Uber_Distance: message protocol used between the service agent Uber Distance and the MCDM agents;

All this message types are exchanged in the system during the negotiation process and the case-based reasoning process.

4.1.7 CBR and Multi-Criteria Decision Making

In this subsection will be detailed the negotiation process that occurs inside the system. The solution uses a combination of Multi-Criteria Decision Making and CBR to achieve restaurant

recommendations that will be detailed next. Figure 21 shows an overview of exchanged message flows within the system.

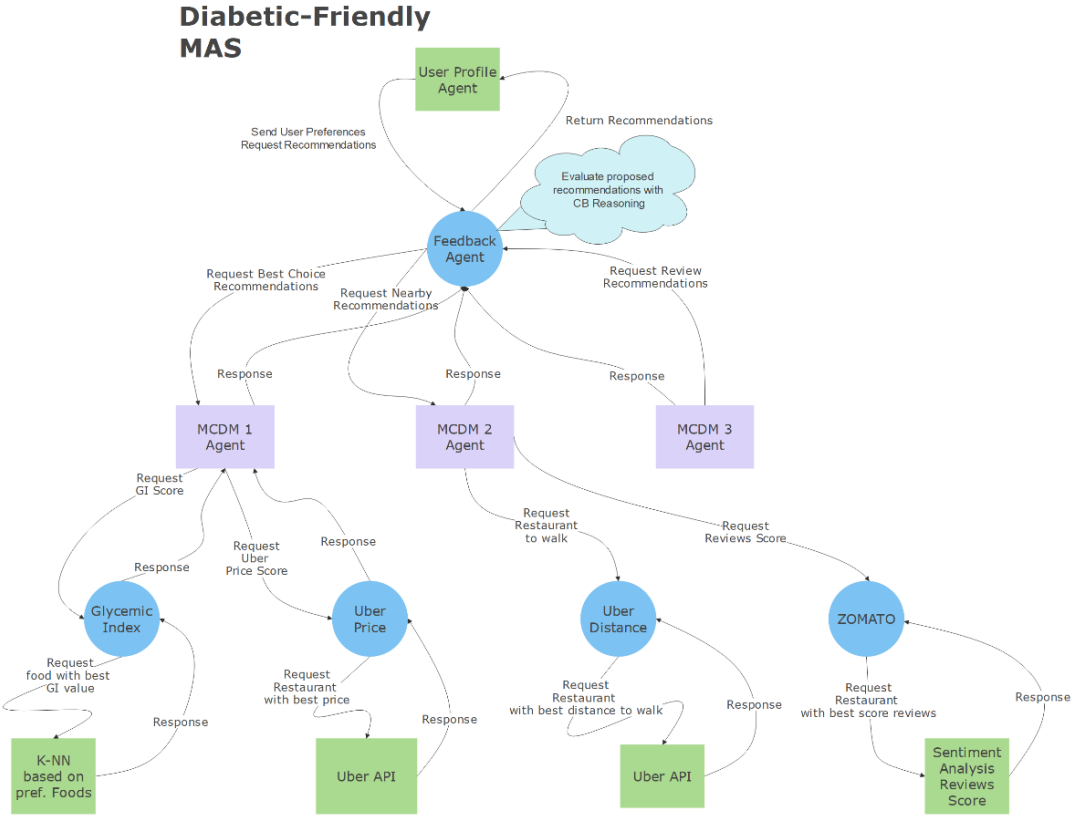

Figure 21 Diagram with exchanged messages between agents

The negotiation process is always initiated by the Feedback User Agent, the one that acts as a broker in the negotiation process. This agent carries the user preferences into the system and perform the initial search for restaurant recommendations. The proposed solution includes three categories of restaurant recommendations (Best Choice, Nearby and Reviews), as identified previously in Table 1. After this search all the MCDM agents will reply to the Feedback User Agent after receiving proposals from the service agents (Glycemic Index Agent, Uber Price Agent, Uber Distance Agent and Reviews Agent).

In the following subsection will be described the specific processes that are implemented for two of the main agents involved in the negotiation, the MCDM agents and the Feedback User Agent. The MCDM Agent will enable the system to provide a set of criteria weights, that will be evaluated inside the Feedback Agent by using the CBR.

4.1.7.1 MCDM Agent

This agent type holds some specific characteristics. The first one is that in the initial agent configuration it is possible to define a “cognitive profile”, this means, the agents can be setup to search for specific recommendation types, or simply ignore them. This enables the existence of multiple agents of this type to exist in the system running at the same time, being ones more fit and eager for specific recommendations than others. This also gives the system more complexity on the negotiation level. Another important feature of this agents is the wake up

behaviour, that is triggered after the negotiation process starts. At current stage the behaviour wakes up after three seconds but could be changed for a generated number between a specific interval, suppose between one and five seconds, giving also more complexity to the negotiation process, that will try to negotiate with the different service agents in different moments in time. This mechanism was created to define a limit time for the recommendation received from the service agents. The algorithm assumes that if the MCDM agent is not capable of receive proposal recommendation in a timely fashion, then it will be discarded by the Feedback User Agent.

```
private void RequestServiceBestChoice() {

    //Glycemic_Index_Score_By_Food
    ServiceDescription sRequest = new ServiceDescription();
    sRequest.setType(MessageProtocol.Glycemic_Index_Score_By_Food.toString());
    SearchForService(sRequest, MessageProtocol.Glycemic_Index_Score_By_Food);

    sRequest = new ServiceDescription();
    sRequest.setType(MessageProtocol.Uber_BestPrice.toString());
    SearchForService(sRequest, MessageProtocol.Uber_BestPrice);

    /**
     * Add the WakerBehaviour (wakeup-time 10 secs)
     * After starting negotiation process, the agent waits 3 secs for proposals from MCDM agents,
     * after the 3 secs it will evaluate proposals with Case Based Reasoning and deliver recommendations to user
     * or not
     */

    Bruno Teixeira (ISEP)
    addBehaviour(new WakerBehaviour( a: this,  timeout: 3000) {
        Bruno Teixeira (ISEP)
        protected void handleElapsedTimeout()
        {...}
    });

}
```

Figure 22 Wake up behavior within MCDM agent

Every time the MCDM agent “wakes up”, it will review the received proposals and can automatically discard the ones that are not compliant with the user preferences, provided as required by the Feedback User Agent on the initial request. This provides a filter and gives the chance for the service agents to provide new restaurant recommendations, if still within the limit time.

The multi-criteria problem, it’s the reviewing of possible restaurant recommendations provided by the service agents to the MCDM agents. Our multi-criteria problem is defined as follows:

Multi-Criteria_Problem = MCP (R,C,W) where:

R = Restaurant Recommendations for user

C=criteria=[Review, Glycemic Index, Price, Distance]

W=weighting restaurant recommendations=[Recommended Review Value, Recommended GI Value, Recommended Price, Recommended Distance]

To solve this multi-criteria decision problem a simple WSM (Weighted Sum Model) will be used. This is one of the best known methods in Multi-Criteria Decision Analysis and given the lack of time to present the proposed solution this was the chosen method. In order to achieve our

utility function, we had to define a decision matrix, that will allow the MCDM agents to select the best alternative from all the alternatives provided by the service agents.

Assuming for instance three alternatives choices A_1, A_2, A_3 each described in terms of our four criteria Review, Glycemic Index, Price and Distance, the following decision matrix is presented:

Table 2 Decision Matrix example

	Review	Glycemic Index	Price	Distance	WSM Score
Weighting	0.25	0.25	0.25	0.25	-
Choice A_1	25	20	20	30	23.75
Choice A_2	10	30	20	30	22.50
Choice A_3	30	10	30	10	20.00

In our scenario all our four criteria have the same weight which is equal to 0.25 or 25%. When the formula is applied on these numerical data the WSM scores for the three alternatives are:

$$A_1^{WSM-score} = 25 \text{ x } 0.25 + 20 \text{ x } 0.25 + 20 \text{ x } 0.25 + 30 \text{ x } 0.25 = 23.75 \qquad (7)$$

Similarly, the other alternative WSM scores are calculated in order to get the best choice, in this case a maximization case, meaning that from the three alternatives the following raking is set as $A_1 > A_2 = A_3$ (where the symbol ">" stands for "greater than"). After selecting the best alternative choice the MCDM Agent will constitute a case, or a valid restaurant recommendation, that will be further evaluated by the CBR algorithm that exists within the Feedback User Agent. The decision model proposed is defined by the following steps:

1. MCDM Agents receive from the Feedback Agent the user criteria definition, which in this case correspond to the user preferences;

2. MCDM Agents request in a timely fashion all the service agents for the values for the four identified weights;

3. Each time the MCDM Agent receive a value from the service agents, it will check if its complying with the criteria received, and if not, the proposal will be discarded, and the service agent can attempt to send a new one;

4. After received valid weights, the MCDM Agent will create use the WSM method to find the best choice and a new case, or possible restaurant recommendation will be sent as a proposal to the Feedback Agent.

In Table 3 are defined all the classes that characterize one case, being a case in this scenario the record containing a potential restaurant recommendation that will be evaluated by the Feedback Agent through the CBR algorithm.

Table 3 Characteristics of each case.

Representation	Name of variable	Variable type	Measurement scale (converted scale)
X1	Id	Numerical	Integer
X2	Recommendation Category	String	String
X3	User Pref - Zomato Review	Numerical	Real Number
X4	User Pref - Glycemic Index	Numerical	Real Number
X5	User Pref - Uber - Price	Numerical	Real Number
X6	User Pref - Uber - Distance	Numerical	Real Number
X7	Recommended Zomato Review	Numerical	Real Number
X8	Recommended Glycemic Index	Numerical	Real Number
X9	Recommended Uber - Price	Numerical	Real Number
X10	Recommended Uber - Distance	Numerical	Real Number
X11	User Action	Numerical	Integer
X12	Created	String	String
R	Similarity Case Score	Numerical	Real Number

4.1.7.2 Feedback User Agent

As previously mentioned, this agent will act as a broker in the negotiation process on the users behalf. After receiving the initial request from the User Agent for recommendation, this agent will call the start negotiation method as Figure 23 illustrates.

```
private void StartNegotiation() {

    //REQUEST MCDM's available to retrieve their best

    //SEARCH FOR MCDM's that can retrieve recommendations for BEST CHOICE
    ServiceDescription serviceRequest = new ServiceDescription();
    serviceRequest.setType(MessageProtocol.Best_Choice.toString());
    SearchForService(serviceRequest, MessageProtocol.Best_Choice);

    //SEARCH FOR MCDM's that can retrieve recommendations for NEARBY
    serviceRequest = new ServiceDescription();
    serviceRequest.setType(MessageProtocol.NearBy.toString());
    SearchForService(serviceRequest, MessageProtocol.NearBy);

    //SEARCH FOR MCDM's that can retrieve recommendations for REVIEWS
    serviceRequest = new ServiceDescription();
    serviceRequest.setType(MessageProtocol.Review.toString());
    SearchForService(serviceRequest, MessageProtocol.Review);

}
```

Figure 23 Feedback User Agent start negotiation method

During the negotiation process the Feedback User Agent will receive multiple responses from the MCDM agents, being a response here a potential restaurant recommendation. Internally the Feedback User Agent holds an evaluation process that takes in consideration previous cases. To enable this evaluation a case-based reasoning algorithm is implemented within the agent. The implementation of the case-based reasoning (CBR) inside the multi-agent system was done by using the jCOLIBRI framework [116]. With this framework was able to manage a database of cases and quickly implement the four steps in the algorithm (Retrieve, Reuse, Revise and Retain).

The CBR process has a main task: the matching. It will search for the *k* closest cases of the proposed case by using similarity measure. The *k-NN* method is already implemented in the jCOLIBRI framework. The process will select the closest or similar case as the following pseudo algorithm shows.

Pseudo algorithm: CBR Process

```
1:  Input: Uhc_RR
2:  NC
3:  Initialize k
4:  Retrieve(NC,Closest_Case) using k-nn
5:  If Revise(CN,Closest_Case) similarity > 0.8
        Send_User_Recommendation(NC, result)
    Else
        Discard_NC
    EndIf
6:  Retain(result,NC,Cls_RR)
```

Uhc_RR: User history cases
NC: new case

The main steps are next described in more detail.

Retrieve. The retrieve step is the most important task of the CBR cycle, and it is the task in which the system selects the most similar case. In the proposed solution, this process employs the *k-NN* technique [80], which is utilized to select the most similar cases. For this purpose, the k-NN algorithm uses a distance measure to analyze each case. This measure is the Euclidean distance and is expressed as:

$$d\left(u_i, u_j\right) = \sqrt{\sum_{r=1}^{n}\left((ui) - (uj)\right)^2}, \quad (8)$$

Where:

n - dimensionality of the input vector, namely the number of attributes of the examples.

r - from 1 to n.

When $d\left(u_i, u_j\right)$ becomes smaller, it means that the two examples are more similar. Equation (9) expresses the prediction that will be the class, and that has the most member in the *k-NNs*

$$y\left(d_i\right) = \arg\max \sum_{u_j \in NN} y\left(u_j, c_k\right) \quad (9)$$

Where:

d_i - text example.

u_j - one of its k-NN in the training.

u_j, c_k - indicates whether u_j belongs to class c_k.

Reuse. Even though this is one of the common steps in the CBR, the implementation proposed is not applying it, since this is optional in the domain considered.

***Revise*.** The revise step validates the similarity value between the new case and the best similar case inside the data base should respect the formula (10). If the new case has a similarity value superior to 80%, then the case is considered valid, and the restaurant recommendation is sent to the user for approval or rejection. If the case is inferior to this value, it will be automatically discarded and the Feedback Agent will notify the MCDM Agent to propose a new case.

$$Similarity\{new\ case, best\ similar\ case\ in\ data\ base\} \leq 80\%, \quad (10)$$

Retain. The retain step is only processed after the result obtained from the user interaction, that could be either an “Accept” or “Reject”. In both scenarios, the step will retain the restaurant recommendation in the database of cases for that user.

4.1.8 API

In this subsection will be described the API built, that allow the Recommender System Module to communicate with the User Interface Module.

This small API contains multiple controllers to handle the exchanged message, that are by definition JSON messages [117]. One example of this controllers is illustrated in Figure 24 with methods to handle the user response to previous provided restaurant recommendations. The methods are respectively for the accept and reject options present for each restaurant recommendation present in the User Interface Module. The main function of this methods is to interact again with the multi-agent system, in order to complete the last phase of the CBR algorithm, the retain step, where the user reply will be saved as a new case in the user case history log.

```
@RestController
public class BestChoiceController {

    Bruno Teixeira (ISEP)
    @CrossOrigin(origins = "http://127.0.0.1:8000/")
    @RequestMapping("/api/AcceptBestChoice")
    @ResponseBody
    public String AcceptBestChoice(HttpEntity<String> httpEntity) throws JSONException
    {...}

    Bruno Teixeira (ISEP)
    @CrossOrigin(origins = "http://127.0.0.1:8000/")
    @RequestMapping("/api/RejectBestChoice")
    @ResponseBody
    public String RejectBestChoice(HttpEntity<String> httpEntity) throws JSONException
    {...}
}
```

Figure 24 API Rest Controller

4.1.9 Reviews Agent and Sentiment Analysis

In this subsection will be described the implementation of the sentiment analysis for restaurant reviews that is included within the Reviews Agent Service.

As part of this implementation the following methods are applied:

1. Pre-process text reviews by making all words lowercase.
2. Apply tokenization.
3. Remove numbers and punctuation, stop words and lemmatization.
4. Create word-to-vector with TD-IDF [118] to have a rating.

Since this analysis intends only to achieve positive, negative, and neutral reviews scores, the following scale was defined:

1. Positive review – will have a rating above 3;
2. Negative reviews will have a rating below 3;
3. Neutral reviews will have a rating of 3.

The metrics used to determine random forest classifiers are precision, recall and accuracy. The list of words that affect the results are {“bad”, ”good”, ”average”, ”best”, ”place”, ”love”, ”order”, ”food”, ”try”, ”nice”]}.

Next, are presented some extracts of the code implementation.

```
def preProcessing():
    tokenized_doc = []
    lemmatized_doc = []
    for d in file_docs:
    sentence = d.lower()

    # Remove stop words - High frequency n-grams
    # sentence = STOPWORDS_RE.sub("", sentence)
    sentence = [word for word in sentence.split() if word not in
stoplist]

    # Lemmatization
    lem = lemSentence(sentence)
    lemmatized_doc.append(lem)

    # Tokenization
    token_words = word_tokenize(lem)
    tokenized_doc.append(token_words)
    return tokenized_doc
```

After the pre-processing phase the following model implementation is called in order to get the reviews ranking.

```
def model(normalized_doc, input_phrase):

    # Create a bag of words
    dictionary = gensim.corpora.Dictionary(normalized_doc)
    # Convert doc to a matrix
    corpus = [dictionary.doc2bow(gen_doc) for gen_doc in
normalized_doc]

    # TD-IDF
    # words that occur more frequently across the reviews get smaller
weights
    tf_idf = gensim.models.TfidfModel(corpus)

    # Creating similarity measure object
    sims = gensim.similarities.Similarity(current_dir, tf_idf[corpus],
num_features = len(dictionary))

    # Create query
    query_doc = [w.lower() for w in word_tokenize(input_phrase)]
    # update an existing dictionary and create bag of words
    query_doc_bow = dictionary.doc2bow(query_doc)
    # perform a similarity query against the corpus
    query_doc_tf_idf = tf_idf[query_doc_bow]

    sims = sims[query_doc_tf_idf]
    sims = sorted(enumerate(sims), key=lambda item: -item[1])

    score = 0
    for doc_position, doc_score in sims:
    if score < doc_score and doc_score != 0 :
    score = doc_score
```

```
    output = file_docs[doc_position]
    return output
```

4.2 User Interface Module

In the current subsection is detailed the User Interface Module (UI Module). As previously mentioned, this module is a web-based portal built in Django technology.

This framework was selected because of the following identified advantages:

- It's a high-level Python web framework;
- Allow rapid development and clean design;
- Holds an integration layer for WebSockets, through the Django Channels, which extends HTPP abilities [119].

Those are not the only advantages of this framework, but those are the ones in-line with the initial idea for the implementation. Since the multi-agent system was coded in Java, the need for a quick way to create a communication between the two modules (UI Module and Recommender System Module) was a critical goal for the success of this solution. The idea behind was to display all the exchanged messages between the agents during the negotiation process to achieve recommendations, to be displayed in real-time on the user interface, without the need to perform any page refresh. WebScokets are perfect fit for this feature and since the Django framework was also a Phyton framework, that was also an important factor, since much of the artificial intelligence models are built with Phyton language nowadays, increasing the potential compatibility during the development process.

The User Interface Module is a web-based portal containing the initial survey for user preferences setup. It is also possible to see a system log message provided by the WebScokets, showing all the messages exchanged by each agent in the system during the negotiation process. In Figure 25 below is shown the user interface, with multiple zones that will be next described.

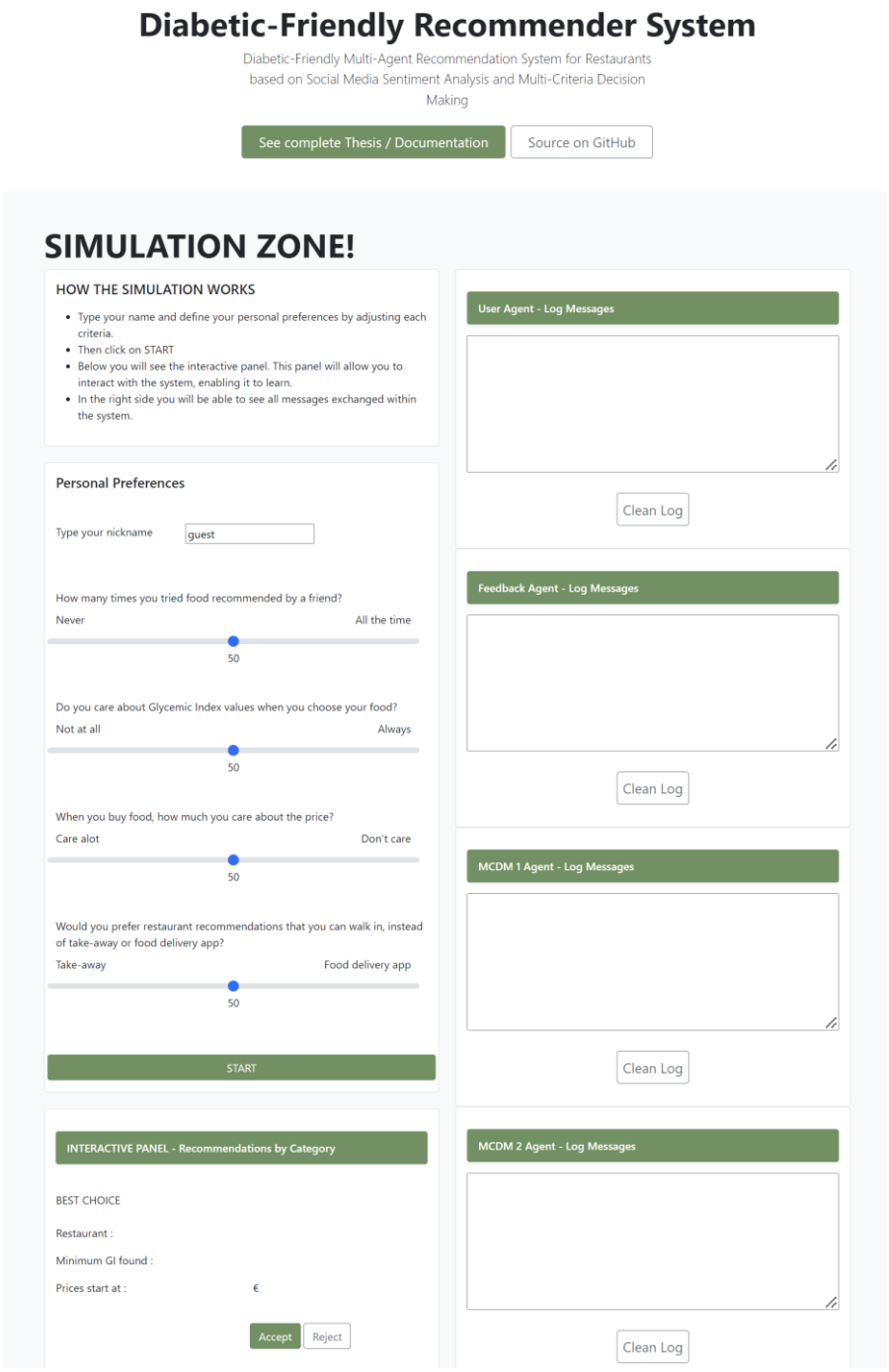

Figure 25 User Interface

The portal contains multiple zones of interaction or visualization. The personal preferences zone as shown in Figure 26 is where the user performs the initial interaction with the system. He can type a nickname that will identify the user in the system, next it will adjust the values of each question of the survey that define the user preferences for the cold start. All the values are in a scale of integer between 0 to 100.

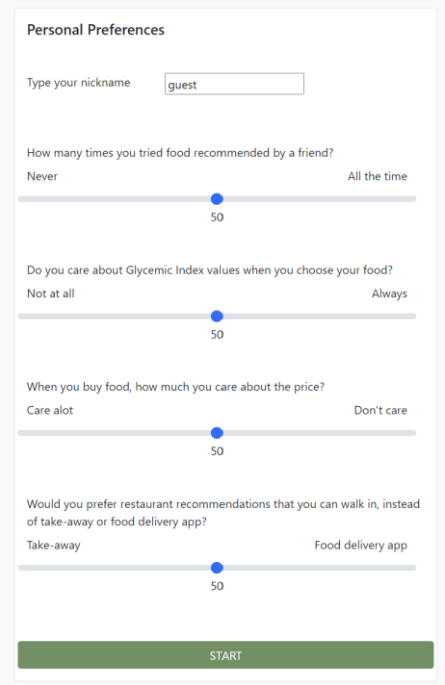

Figure 26 UI Personal Preferences

Once the user hits the “Start” button, the Recommender System Module is notified to start the negotiation process and during that period the user will start to see the message exchange by each agent in the agent’s message log panels. Figure 27 shows an example of messages associated with one of the MCDM agents in the system.

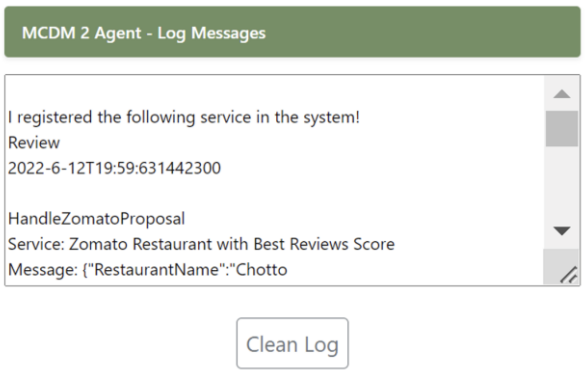

Figure 27 UI Agent Log Message

After the negotiation process to occur, the user will start receiving restaurant recommendation in the interactive panel zone. This zone displays the three categories defined for the proposed solution and details about the respective recommendation.

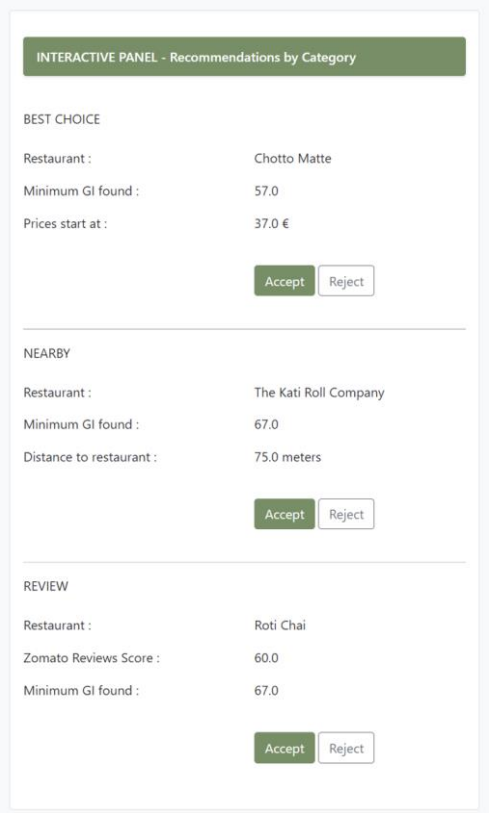

Figure 28 UI Interactive Panel

As illustrated by Figure 28, the user will have the ability to accept or reject each individual restaurant recommendation. These actions will trigger the call of methods that exist in the API built for the Recommender System Module. The reject feature also indicates the RS Module to perform a new round of negotiation, so that a new restaurant recommendation could be provided.

5 Results and Discussion

In this chapter will be described the process of experimentation and evaluation of the proposed solution, being presented the hypothesis tested and the respective methodology for evaluating. At the end of the chapter are presented and discussed the results obtained.

5.1 Hypotheses

In order to carry out a correct evaluation of the final solution, it is necessary, first of all, to define specifically what will be evaluated. For that, hypotheses will be tested during the assessment process. Having as main objective to provide the user with recommendation, it is fundamental for the proposed work to evaluate if those are correctly generated. As mentioned before, the generation of this recommendations is evaluated with the use of the CBR algorithm. Therefore, the validation of the developed solution consists in the study of the impact of the use or not of the CBR. So, the hypothesis to validate is defined as:

- The use of the CBR algorithm will provide more accurate restaurant recommendations, compared with those provided without the use of the CBR algorithm.

To test the defined hypothesis, two scenarios were created as described in more detail in the following subsections.

5.2 Scenario I

The Scenario I contemplate a set of ten simulations. Those were made using the CBR algorithm and then ten different simulations were performed without CBR. Also, these simulations were running under pre-defined set of cases along with the preferences for the user with middle values defined. The only random values were generated by the service agents, when MCDM agents request new recommendation options. The current service agents provide values for Recommended Reviews, Recommended Uber Price, Recommended Uber Distance and Recommended Glycemic Index. All those values are currently being generated randomly and between 0 and 100 since that was the pre-defined scale. The following table shows the similarity score percentage between the recommended case and the database cases for that user, by recommendation category. Those values test the application of the CBR algorithm.

Table 4 Simulation results for Scenario I with CBR algorithm

Recommendation Category	Restaurant Name	Similarity Score
Best Choice	Tayyabs High	79,50%
Best Choice	Tayyabs Low	91,67%
Nearby	The Kati Roll Company High	100,00%
Nearby	The Kati Roll Company Low	98,97%
Reviews	Roti Chai High	77,88%
Reviews	Roty Chai Low	95,80%

All the results using the CBR algorithm as we can see in Table 4 obtain a high result in terms of similarity, with all results with values superior to 75% for all the recommendation categories considered. After those initial 10 simulations with CBR, another ten simulations were performed without CBR algorithm application.

Table 5 Simulation results for Scenario I without CBR algorithm

Recommendation Category	Restaurant Name	Similarity Score
Best Choice	Tayyabs High	64,85%
Best Choice	Tayyabs Low	77,45%
Nearby	The Kati Roll Company High	46,95%
Nearby	The Kati Roll Company Low	53,78%
Reviews	Roti Chai High	45,06%
Reviews	Roti Chai Low	51,85%

The results obtained in Table 5 are much lower in similarity score, with the worst value being only 45,06%. The following table provides an overview of both results, were clearly we can see a better performance, when CBR algorithm is applied.

Table 6 Comparison between similarity score results with and without CBR

Restaurant Name	Similarity Score with CBR	Similarity Score without CBR
Tayyabs High	79,50%	64,85%
Tayyabs Low	91,67%	77,45%
The Kati Roll Company High	100,00%	46,95%
The Kati Roll Company Low	98,97%	53,78%
Roti Chai High	77,88%	45,06%
Roti Chai Low	95,80%	51,85%

5.3 Scenario II

For the second simulation scenario, only ten simulations were performed, but in this case both similarity scores were taken at each simulation. This second scenario confirms also that when a CBR algorithm is applied, the system performance is greater.

Table 7 Simulation II results with and without CBR algorithm

Recommendation Category	Case	Similarity Score with CBR	Similarity Score without CBR	Recommended Restaurant
Best Choice	1	0,911764706	0,746573898	Chotto Matte BP Low
Nearby	1	1	0,52294686	The Kati Roll Company Low
Reviews	1	0,955223881	0,53814262	Roti Chai Low
Best Choice	2	0,987179487	0,773504274	Tayyabs Low
Nearby	2	1	0,457207207	The Kati Roll Company High
Reviews	2	0,927927928	0,529279279	Roti Chai Low
…	…	…	…	…
Best Choice	10	0,857212476	0,682423652	Chotto Matte BP Low
Nearby	10	1	0,509615385	The Kati Roll Company High
Reviews	10	0,660266257	0,362854461	Roti Chai High
	Average Score	91,84%	56,94%	

After this second set of simulation, we obtained an average score of 91,84% using CBR and 56,94% without CBR, which clearly confirms the advantage of using this technique as part of the solution. Our hypothesis proves in this case to be correct and confirmed by evaluating the similarity score. This value indicates how close the recommendations provided by the system are more in line with the user preferences when the CBR algorithm is used.

6 Conclusion and Future Work

6.1 Synthesis and conclusion of the work developed

The proposed solution contemplates the development of a recommender system, based on multi-agent systems that implement a set of artificial techniques to allow a diabetic-friendly feature that will recommend the user with healthier diet restaurants with meals that potentially improve quality of life and have health benefits.

The system is currently working as expected, allowing to perform different simulations. The work is clearly unfished since the solution propose the framework base, which can be improved and other types of simulation can be performed to further validate the solution. As future work some key points have already been identified that will turn the solution more robust. The complexity of the solution can be increased, especially in the negotiation process, where new types of service agents could be added to the system, turning the multi-criteria decision making a much more powerful component in the solution. One of the biggest advantages of building the solution based on multi-agent system is that new service agents can be added to the system without changes to the base of the framework. Still related with the service agents, there is the potential to improve the Glycose Index agent, by adding for instance object detection functionalities that will allow the user to capture an image of a meal, so that presented food is automatically recognized, and the corresponding Glycemic Index values are measured, making the recommendation done by this agent much more accurate. The solution could also be incorporated within an application like Uber Eats, Zomato or Trip Advisor as a new diabetic-friendly feature in the same sense we can see today for example for vegetarian people. Other methodologies could also be tested and finally, real case scenarios should also provide validation and the applicability of this solution for diabetic people in real life.

6.2 Scientific Impact

During the development of this work a paper was written and accepted to be published for the 21st EPIA Conference. The EPIA Conference on Artificial Intelligence is a well-established European conference [120], that will take place at Instituto Superior Técnico between August 31st and September 2nd, 2022. The paper presents the architecture proposed for this solution and the current dissertation includes the simulations performed to valid the use of the CBR approach.

Bibliographic Reference

[1] WHO, "Diabetes," 10 11 2021. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes.

[2] "Diabetes | Factos e Números 2015," Serviço Nacional de Saúde, 15 03 2017. [Online]. Available:https://www.sns.gov.pt/noticias/2017/03/15/diabetes-factos-e-numeros-2015/.

[3] C. Awuchi et. al, "Diabetes and the Nutrition and Diets for Its Prevention and Treatment: A Systematic Review and Dietetic Perspective," *Health Sciences Research,* pp. 5-19, 2020.

[4] D. Martinho et al, "A systematic review of gamification techniques applied to elderly care," *Artif Intell Rev,* 2020.

[5] WHO, "Diabetes Fact Sheet Nº312," 2013.

[6] D. Gardner et al, Greenspan's basic & clinical endocrinology (9th ed.), New York: McGraw Hill Medical, 2011.

[7] A. Norman et al, "Hormones. Elsevier," in *Hormones. Elsevier*, 2015, pp. 136-137.

[8] J. Irwin, "Manual of intensive care medicine (5th ed.)," em *Manual of intensive care medicine (5th ed.)*, 2010, p. 549.

[9] J. Cash et al, "Family Practice Guidelines (3rd ed.)," 2014, p. 396.

[10] C. G. Awuchi, "Sugar Alchohols: Chemistry, Production, Helath Concerns and Nutritional Importance of Mannitol, Sorbitol, Xylitol and Erythritol," *International Journal of Advanced Academic Research,* pp. 30-66, 2017.

[11] I. D. Federation, "IDF Diabetes Atlas, 8th," Brussels, Belgium, 2017.

[12] S. Yuankai et al, "The global implications of diabetes and cancer," p. 383, 2014.

[13] CDC, "Deaths and Cost | Data & Statistics | Diabetes," *Deaths and Cost | Data & Statistics | Diabetes,* February 2019.

[14] H. Kyu et al, "Physical activity & risks of breast cancer, colon cancer, diabetes, ischemic heart disease & ischemic stroke events: systematic review & dose-response meta-analysis," 2013.

[15] P. Health et al, "The Nutrition Source," *The Nutrition Source,* 2012.

[16] C. Willi et al, “Active smoking & the risk of type 2 diabetes: systematic review & meta-analysis,” *Active smoking & the risk of type 2 diabetes: systematic review & meta-analysis,* pp. 2654-2664, 2007.

[17] M. Nathan et al, “Intensive diabetes treatment & cardiovascular disease in patients with type 1 diabetes,” *The New England Journal of Medicine,* pp. 2643-53, 2005.

[18] A. Qaseem et al, “Hemoglobin A1c Targets for the Glycemic Control With the Pharmacologic Therapy for Nonpregnant Adults with Type 2 Diabetes Mellitus: Guidance Statement Update From American College of Physicians,” in *Annals of Internal Med.*, 2018, pp. 569-576.

[19] P. Cavanagh, “Therapeutic footwear for people with diabetes,” 2004, pp. 51-55.

[20] J. Haw et al, “Longterm Sustainability of the Diabetes Prevention Approaches: A Systematic Review and Meta-analysis of Randomized Clinical Trials,” in *JAMA Internal Medicine*, 2017, pp. 1808-17.

[21] A. Mottalib et al, “Weight Management in Patients with the Type 1 Diabetes & Obesity,” Current Diabetes Reports. Vol 17 (10): p 92. doi: 10.1007/s11892-017-0918-8, 2017.

[22] A. D. Association, “5. Lifestyle Management: the Standards of Medical Care in Diabetes,” em *The Diabetes Care*, 2019.

[23] A. Evert et al, “Nutrition Therapy for the Adults with Diabetes or Prediabetes: Consensus Report,” in *Diabetes Care (the Professional society guidelines). Vol 42*, (2019)., p. 731–54.

[24] A. Emadian et al, “The effect of macronutrients on the glycaemic control: a systematic review of the dietary randomized controlled trials in overweight & obese adults with type 2 diabetes in which there was no difference in weight loss between treatment groups,” *The British Journal of Nutrition,* p. 1656–66, 2015.

[25] “ITEA 4 Food Friend,” [Online]. Available: https://itea4.org/project/food-friend.html. [Last access in 20 06 2022].

[26] B. Kantor et al, Recommender Systems Handbook, 2011.

[27] G. Ziegler et al, “Improving recommendation lists through topic diversification,” in *Proceedings of the 14th international conference on World Wide Web*, 2005.

[28] C. Kadie et al, “Empirical analysis of predictive algorithms for collaborative filtering,” in *In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI'98)*, 1998.

[29] C. Kadie et al, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” 1998.

[30] S. Szedmak et al, “Kernel-Mapping Recommender system algorithms,” *Information Sciences,* pp. 81-104, 2012.

[31] J. Riedl et al, “Application of Dimensionality Reduction in Recommender System A Case Study,” 2000.

[32] R. Allen, “User Models: Theory, Method, Practice,” em *International J. Man-Machine Studies*, 1990.

[33] K. Gallagher et al, “Using viewing time to infer user preference in recommender systems,” in *AAAI Workshop in Semantic Web Personalization*, San Jose, California, 2004.

[34] Y. Park et al, “Discovery of Hidden Similarity on Collaborative Filtering to Overcome Sparsity Problem,” *Discovery Science,* 2007.

[35] C. Felício et al, “A Multi-Armed Bandit Model Selection for Cold-Start User Recommendation,” in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization. UMAP '17*, 2017.

[36] ChenPu et al, “Differentiating Regularization Weights -- A Simple Mechanism to Alleviate Cold Start in Recommender Systems,” in *ACM Transactions on Knowledge Discovery from Data (TKDD)*, pp. 1-22.

[37] D. Kaplan et al, “Active Learning in Recommender Systems,” in *In Ricci, Francesco; Rokach, Lior; Shapira, Bracha (eds.). Recommender Systems Handbook (2 ed.).*, 2016.

[38] M. Elahi et al, “A survey of active learning in collaborative filtering recommender systems,” in *Computer Science Review*, 2016, pp. 29-50.

[39] X. Bi et al, “A group-specific recommender system,” *Journal of the American Statistical Association.,* pp. 1344-1353, 2017.

[40] G. Linden et al, “Collaborative Recommendations Using Item-to-Item Similarity Mappings,” Wayback Machine, 2015.

[41] B. Shapira et al, “Introduction to Recommender Systems Handbook,” in *Recommender Systems Handbook*, 2011, pp. 1-35.

[42] C. C. Aggarwal, Recommender Systems: The Textbook., 2016.

[43] P. Brusilovsky, The Adaptive Web, 2007.

[44] D. Wang et al, “A content-based recommender system for computer science publications,” *Knowledge-Based Systems,* pp. 1-9, 2018.

[45] S. Blanda, “Online Recommender Systems – How Does a Website Know What I Want?,” *American Mathematical Society.,* 2016.

[46] Y. Feng et al, “The Deep Learning–Based Recommender System “Pubmender” for Choosing a Biomedical Publication Venue: Development and Validation Study,” *Journal of Medical Internet Research,* 2019.

[47] B. Hidasi et al, “Session-based Recommendations with Recurrent Neural Networks,” 2016.

[48] M. Chen et al, “Top-K Off-Policy Correction for a REINFORCE Recommender System,” 2018.

[49] M. Yifei et al, “Temporal-Contextual Recommendation in Real-Time,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery*, 2020.

[50] B. Hidasi et al, “Recurrent Neural Networks with Top-k Gains for Session-based Recommendations,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management. CIKM '18*, 2018.

[51] W. Kang et al, “Self-Attentive Sequential Recommendation,” *Self-Attentive Sequential Recommendation,* 2018.

[52] J. Li et al, “Neural Attentive Session-based Recommendation,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. CIKM '17*, 2017.

[53] Q. Liu et al, “STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '18*, 2018.

[54] X. Xin et al, “Self-Supervised Reinforcement Learning for Recommender Systems,” 2020.

[55] C. Boutilier et al, “SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets,” in *Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence*, 2019.

[56] L. Zou et al, “Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems,” in *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[57] D. Bouneffouf, “DRARS, A Dynamic Risk-Aware Recommender System,” Institut National des Télécommunications, 2013.

[58] Y. Ge et al, “An Energy-Efficient Mobile Recommender System,” in *Proceedings of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2010.

[59] E. Pimenidis et al, “Mobile recommender systems: Identifying the major concepts,” *Journal of Information Science,* pp. 387-397, 2018.

[60] R. Hoekstra, “The Knowledge Reengineering Bottleneck,” *Semantic Web – Interoperability, Usability, Applicability,* 2010.

[61] C. Gomez-Uribe et al, “The Netflix Recommender System,” in *ACM Transactions on Management Information Systems*, 2015, pp. 1-19.

[62] Z. Darban et al, “GHRS: Graph-based hybrid recommendation system with application to movie recommendation,” in *Expert Systems with Applications*, 2022.

[63] R. Burke, “Hybrid Web Recommender Systems,” in *Wayback Machine*, 2014.

[64] C. Kahraman et al, “Fuzzy Multicriteria Decision-Making: A Literature Review,” 2015.

[65] D. Martinho et al, “A web-based group decision support system for multicriteria problems,” *Concurrency Computation,* 2021.

[66] J. Carneiro et al, “Representing decision-makers using styles of behavior: An approach designed for group decision support systems,” in *Cognitive Systems Research*, 2018.

[67] J. Carneiro et al, “Dynamic argumentation in UbiGDSS,” 2018.

[68] J. Carneiro et al, “A General Template to Configure Multi-Criteria,” *International Journal of Software Engineering and its Applications,* 2015.

[69] D. Martinho et al, “A Multiple Criteria Decision Analysis Framework for Dispersed Group Decision-Making Contexts,” *Applied Sciences,* 2020.

[70] T. Saaty, “The Theory of Ratio Scale Estimation: Saaty's Analytic Hierarchy Process,” *The Analytic Hierarchy Process.,* 1980.

[71] J. Carneiro et al, “Including cognitive aspects in multiple criteria decision analysis,” *Ann Oper Res,* 2018.

[72] P. Fishburn, “Additive Utilities with Incomplete Product Set: Applications to Priorities and Assignments,” *Journal of the Operations Research Society of America,* 1967.

[73] A. Aamodt et al, “ Case-Based Reasoning: Foundational Issues, Methodological Variations, and System,” in *AI Communications.*, 1994, pp. 39-59.

[74] P. Barrett, Euclidean distance: raw, normalized, and double-scaled coefficients, 2005.

[75] T. Phyu, Survey of Classification Techniques in Data Mining. Lect. Notes Eng. Comput. Sci., 2009.

[76] G. Sidorov et al, Soft Similarity and Soft Cosine Measure:, 2014.

[77] A. Huang, “Similarity measures for text document clustering,” in *Proc. 6th N. Z. Comput. Sci. Res. Stud. Conf.*, 2008.

[78] “The Cosine Similarity algorithm - Chapter 8. Similarity algorithms,” 2019. [Online]. [Last access in 20 06 2022].

[79] “Statistics How To,” [Online]. Available: https://www.statisticshowto.com/jaccard-index/. [Last access in 20 06 2022].

[80] Z. Deng et al, “Efficient kNN clas-sification algorithm for bid data,” in *Neurocomputing*, 2016, pp. 143-148.

[81] M. Wooldrige et al, “An Introduction to MultiAgent Systems,” 2009. [Online].

[82] M. Genesereth et al, “Knowledge Interchange Format” 1992.

[83] T. Finin et al, *DRAFT Specification of the KQML Agent-Communication Language,* 1993.

[84] “FIPA,” [Online]. Available: http://www.fipa.org/repository/aclspecs.html. [Last access in 20 06 2022].

[85] R. Nwana et al, “Coordination in Software Agent Systems,” *British Telecommunications Technology Journal,* 1996.

[86] M. Beer, “Negotiation in multi-agent systems,” *The Knowledge Engineering Review,* pp. 285-289, 1999.

[87] M. Jennings et al, “Automated negotiation: prospects, methods and challenges,” *International Journal of Group Decision and Negotiation,* p. 199–215, 2001.

[88] L. Rahwan et al, “Argumentation-based negotiation,” *The Knowledge Engineering Review,* pp. 343-375, 2003.

[89] P. Hadidi et al, “Argumentative alternating offers. Argumentation in Multi-Agent Systems,” pp. 105-122, 2011.

[90] S. Kraus, “Automated negotiation and decision making in multiagent environments,” *Multi-agent systems and applications. Springer Berlin Heidelberg,,* pp. 150-172, 2001.

[91] O. Neumann et al, “Theory of games,” *Princeton university press.,* 2007.

[92] A. Kraus et al, “Reaching agreements through argumentation: a logical model and implementation,” *Artificial Intelligence,* pp. 1-69, 1998.

[93] J. Carneiro et al, “Defining Agents’ Behaviour for Negotiation Contexts,” in *Progress in Artificial Intelligence. EPIA 2015*, 2015.

[94] D. Martinho et al,” *Ambient Intelligence and Smart Environments,* 2015.

[95] J. Carneiro et al, “Intelligent negotiation model for ubiquitous group decision scenarios,” *Frontiers Inf Technol Electronic Eng,* 2016.

[96] T. Mikalai et al, “Survey on mining subjective data on the web,” *Data Min Knowl Discov,* pp. 478-517, 2012.

[97] T. Wilson et al, “Recognizing contextual polarity in phrase-level sentiment analysis,” *Proceedings of HLT/EMNLP,* 2005.

[98] B. Liu, “Sentiment analysis and opinion mining,” *Synth Lect Human Lang Technol,* 2012.

[99] C. Aggarwal et al, “Mining Text Data,” *Science+Business Media, LLC’12,* 2012.

[100] M. Yelena et al, “Exploring feature definition and selection for sentiment classifiers.,” *Proceedings of the fifth international AAAI conference on weblogs and social media,* 2011.

[101] C. Whitelaw et al, “Using appraisal groups for sentiment analysis,” *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM),* p. 625–31, 2005.

[102] A. Diana Maynard, “Automatic detection of political opinions in tweets,” *Proceedings of the 8th international conference on the semantic web, ESWC’11,* p. 88–99, 2011.

[103] A. Adomavicius, Multi-Criteria Recommender Systems, 2010.

[104] S. French et al, “Multicriteria Methodology for Decision Aiding.,” *The Journal of the Operational Research Society,* 1997.

[105] A. Andronico et al, “Integrating a multi-agent recommendation system into a Mobile Learning Management System,” 2003.

[106] A. J. Morais et al, “A Multi-Agent Recommender System,” 2012.

[107] V. N. Marivate et al, “An intelligent multi-agent recommender system for human capacity building,” *Proceedings of the Mediterranean Electrotechnical Conference - MELECON,* 2008.

[108] H. Tang et al, “A survey on sentiment detection of reviews,” *Expert Systems with Applications,* 2009.

[109] A. Bermingham et al, “Classifying sentiment in microblogs: Is brevity an advantage?,” in *International Conference on Information and Knowledge Management, Proceedings*, 2010.

[110] B. Jonathan et al, “Sentiment analysis of customer reviews in zomato bangalore restaurants using random forest classifier,” *Abstract Proceedings International Scholars Conference,* 2019.

[111] S. Trivedi et al, “Twitter sentiment analysis of app based online food delivery companies,” *Global Knowledge, Memory and Communication,* 2021.

[112] “JAVA Agent DEvelopment Framework,” [Online]. Available: https://jade.tilab.com/.

[113] “Django,” [Online]. Available: https://www.djangoproject.com/. [Last access in 20 06 2022].

[114] H. ORM. [Online]. Available: https://hibernate.org/orm/. [Last access in 20 06 2022].

[115] “SQLite,” [Online]. Available: https://www.sqlite.org/index.html. [Last access in 20 06 2022].

[116] “jCollibri,” [Online]. Available: https://gaia.fdi.ucm.es/research/colibri/. [Last access in 20 06 2022].

[117] “JSON,” [Online]. Available: https://www.json.org/json-en.html. [Last access in 20 06 2022].

[118] A. Rajaraman et al, “Mining of Massive Datasets,” in *Data Mining*, 2011, p. 1–17.

[119] “Django Channels,” [Online]. Available: https://channels.readthedocs.io/en/stable/. [Last access in 20 06 2022].

[120] “EPIA Conference,” [Online]. Available: https://epia2022.inesc-id.pt/. [Last access in 20 06 2022].

[121] “Multi-agent systems,” [Online]. Available: https://www.turing.ac.uk/research/interest-groups/multi-agent-systems.

[122] M.Hussein et al, “A survey on sentiment analysis challenges,” *Journal of King Saud University - Engineering Sciences,* 2018.

[123] F. Ricci et al, “Recommender systems: Introduction and challenges,” in *Recommender systems: Introduction and challenges*, 2015.

[124] R. Faia et al, “Dynamic Fuzzy Clustering Method for Decision Support in Electricity Markets Negotiation,” 2016.

[125] [Online]. Available: https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana. [Last access in 20 06 2022].

[126] A. Vieira, “Desenvolvimento de um Sistema para Suporte Individualizado Inteligente a Pacientes com Doença Respiratória Obstrutiva Crónica (Master dissertation),” 2019.