# 3. Theoretical Concepts

This section explains the most relevant theoretical concepts that were applied during the course of this project to provide a fundamental understanding of the application’s main components: chatbot, recommendation and geographic database.

## 3.1 Chatbots

**A chatbot is a computer program that interacts with its user via a chat interface1.** About 60 years ago, the first steps in the development of chatbots were taken by the computer scientists Alan Turing and Joseph Weizenbaum, proposing the concept of computers communicating like humans do*.* One of the first natural language processing programs was ELIZA, developed by Joseph Weizenbaum in 1966. Although some users were tricked into thinking that ELIZA was an actual human conversation partner ², this basic example was stretched to its limit quickly because of its simple rule-based structure. However, the fascination of computers being a conversation partner remained one of the big objectives of modern artificial intelligence.

The topic’s big revival occurred with the introduction of mobile devices in the early 2000s. All of the sudden, developers were faced with the task of transforming their well-known desktop applications and websites into apps to make them suitable for a mobile market. Still, in the last couple of years it turned out that users actually do not like to use a variety of apps, but rather concentrate on only a few, mainly messaging apps. That is how in 2016, the idea of the conversational interface resurged when many global big-players like Google, Facebook, Microsoft, IBM or Amazon decided to take part in the development of conversational interfaces3.

After explaining briefly the importance of chatbots over the years, the following part is going to define the different terms *Natural Language Processing*, *Conversational Interface* and *Chatbots* to outline the differences between them. After that, the key concepts needed to model a conversation flow are introduced.

### 3.1.1 Natural Language Processing, Conversational Interface and Chatbots

Natural Language Processing is a component in the field of Artificial Intelligence in which natural language is analyzed and processed in a way that computers can use converted information easily in further algorithms 3110. Natural Language Understanding is a subdiscipline of the above stated, focusing on “reading comprehension and semantic analysis” 3111.

Nowadays, many big players provide free to use natural language understanding platforms that make the development of conversational interfaces possible. One of these NLP-NLU platforms is api.ai which translates human language into a formal representation using machine learning techniques as well the later explained NLU concepts entities, intents and concepts 3112.

Using a conversational interface, people are enabled to interact with virtual assistants, smart devices and social robot via their natural language 3113­. Conversational interfaces are considered as the third wave of user experience, after the terminal and the graphical user interface. The ideal is that by having a conversational interface, user do not have to adapt to the computer, but the computer has to adapt to the human way of communicating.

There are two basic types of conversational interfaces: voice assistants such as Apple’s Siri or Amazon’s Alexa that communicate using spoken language and chatbots, enabling communication via typing 3114. Basically, in chatbots pattern matching is used to interpret the user’s inputs and templates are used to privde the system’s output. 3113

### 3.1.2 Concepts

The conversational interface used in this project is developed using the NLP-NLU platform api.ai that relies on certain key concepts that are explained in the following.

Using api.ai, an *agent* is modeled to parse the user’s natural input into structured data. In an agent, the conversation flow with the user is specified using the key components entities, intents and context. After designing the agent, it can be integrated into many different platforms and thus providing an application’s conversational interface3121.

The agent relies on *machine learning* algorithms to understand the user input and extract relevant data. Before being confronted with the actual user, input examples are to be specified. Based on these examples, the agent’s machine learning model decides which path of the conversation flow is chosen. As usual in machine learning, the agent learns to adapt better to the user as it is provided constantly with real-life conversations3122.

The main components used in modeling the conversation flow are entities, intents and context. *Entities* are domain objects an application takes actions on. They can be considered as parameters of an action to be taken. In api.ai, there are several already defined system entities, e.g. specifying parameters of time, units and geography. Additionally, the developer can define his own entities. 3123

In an api.ai agent, user requests are mapped to *intents*. By matching the user input with previously specified examples, intents are extracted and used to trigger an action. In our tourist chatbot example, a typical intent would be “Give Recommendation” if the user asks for a tourist recommendation nearby. 3124

When defining an intent, the developer has the option to set an output *context*. Contexts manage the conversation flow by distinguishing the state the conversation is in. Based on the state, the agent may take different decisions on the same user input. 3125

## 3.2 Recommender System

With the advent of business in the Internet, so-called Recommender Systems were introduced and gained importance since the nineties 320. A Recommender System aims to predict and quantify how a user reacts to a certain item. A variety of recommendation algorithms exists, all of them based on collected user preference data (e.g. item ratings). The most famous approaches are *collaborative* and *content-based filtering* that were also used in this project to recommend *Points of Interests* (see 3.3.2) to the users based on their travel interests. In the following, the applied methods are introduced.

### 3.2.1 Content-based filtering

Content-based recommendation systems calculate recommendations based on the properties and characteristics of an item. In general terms, an item is recommended to a user if he is interested in its properties. To define the similarity of items, an item profile is designed representing its important characteristics. In our tourist example, the item profile contains tourist categories extracting data from OSM tags *(more detailed explanations in Relevant Aspects).* Then, user profiles are created containing the same categories as the item profile, indicating which characteristics a user prefers in an item. Usually, this user profile is filled by examining the user ratings and extracting the characteristic for the already rated items.

The preference for a user liking a certain item is then calculated by comparing its profile with the item profiles. To do so, different measures can be applied, one of the most famous being the cosine distance. 321

### 3.2.2 Collaborative Filtering

While the previously presented approach concentrates on the similarity between items for recommendation, collaborative filtering is based on the similarity between users. An item is recommended to a user when similar users have showed an interest in it before. Instead of profiles representing preferences, the only needed data for this approach is the matrix of user ratings. There are different measures to determine if two users are similar weighting user ratings differently, such as the *cosine distance* or the *Jaccard distance*. The collaborative filtering mechanism is very successful as it often provides recommendations outside the expected scope of user interests. However, in order to work correctly, a large amount of user ratings is needed.

### 3.2.1 Hybrid Approaches

Due to the fact that collaborative filtering suffers from the so-called *cold start problem* (that is not performing well when there is only sparse user data), it is often combined with other approaches. One of the most common solutions is falling back to content-based algorithms when user ratings are not significant and hence building up the user rating database. In this project, a hybrid mechanism is applied, although some modifications of the traditional content-based approach were made. The detailed way of proceeding and made adaptions are explained in the Relevant Aspects section of this paper.

## 3.3 Geographic information processing

**Point of Interest**

In the context of geographic information, the expression *Point of Interest 6*  is often used to describe a map feature that has a certain significance. It is a broad term that reaches from functional services like post offices or car parks to tourist attractions. One of this project’s main elements is to extract essential tourist information from the vast amount of geographic data, meaning finding relevant points of interests and outputting them in a comprehensible way for the user.

**Geographic Database Systems**

The geographic information used in this project is stored in a geographic or *spatial* database. While the most common kind of databases nowadays is relational, they are unable to handle geographic data because of its complexity [7]. This is why for this task spatial database systems are used which provide geospatial data types in its data model and query language [8]. The objects in the database contain a spatial or geometric attribute which describes their location, shape, orientation and size.

In this project, geographic information from *OpenStreetMap* is used and stored in a spatial database using *PostGis* (see section 4, Tools and Technologies).

**Extracting Points of Interests from OSM data**

To extract our needed points of interest from the raw geographic data, the underlying data structure of the OpenStreetMap information is examined, consisting of the following four principal elements: [9]

* **Nodes**: Points with a geographic position that are used to represent map features without a size, such as points of interest or mountain peaks.
* **Ways**: Ordered lists of nodes that are used both for representing linear features such as streets and rivers, and areas, like forests, parks, parking areas and lakes.
* **Relations**: Ordered lists of nodes, ways and relations, so-called members. Relations are used for representing the relationship of existing nodes and ways, e.g. long-distance motorways of areas with holes.
* **Tags**: Key-Value pairs storing metadata of the geographic objects they are attached to (namely node, way or relation). Tags can include type, name and a broad variety of map features.

Primarily, the data structures Nodes, Ways and Tags are investigated in detail to construct the points of interest. Like already stated, the *tags* consist of key-value pairs that give conclusions about the attached objects, in this case the nodes and ways. Because of this fact, the tags are examined for keys that would indicate if the nodes or ways are of touristic interest*.* For a detailed explanation of the investigated keys see section 5, Relevant Aspects. Using the examined tags, it is possible to build a query that filters the set of nodes and ways, so only those are returned that match the above mentioned criteria. The second step is to filter the nodes according to their location. Given the fact that the current user location is stated in coordinates (longitude and latitude), only points of interests are examined that are in a certain walking distance from that user.

These requirements lead to the following query:

SELECT ways.id, ST\_Y(geom) as lat, ST\_X(geom) as lon, ways.tags-> 'name' as \_name,

ways.tags-> 'tourism' as tourism, ways.tags-> 'amenity' as amenity,

ways.tags-> 'leisure' as leisure,

ways.tags-> 'cuisine' as cuisine, ways.tags-> 'historic' as historic, ways.tags-> 'shop' as shop,

ways.tags-> 'beach' as beach, nodes.tags-> 'addr:street' as street,

ways.tags->'addr:housenumber' as housenumber,

ways.tags-> 'opening\_hours' as openingHours,

ST\_Distance(geography(geom), ST\_SetSRID(geography(ST\_Point(***longitude, latitude***), 4326)) as distance

FROM ways inner join nodes on ways.nodes[1]=nodes.id WHERE ST\_DWithin(geography(nodes.geom), ST\_SetSRID(geography(ST\_Point(***longitude****,* ***latitude***)), 4326), ***radius***) and ways.tags ? 'name' and ways.tags ?| ARRAY['tourism','amenity','leisure','cuisine','beach','historic','shop']

UNION ALL

SELECT nodes.id, ST\_Y(geom) as lat, ST\_X(geom) as lon, nodes.tags-> 'name' as \_name,

nodes.tags-> 'tourism' as tourism, nodes.tags-> 'amenity' as amenity,

nodes.tags-> 'leisure' as leisure,

nodes.tags-> 'cuisine' as cuisine, nodes.tags-> 'historic' as historic, nodes.tags-> 'shop' as shop,

nodes.tags-> 'beach' as beach, nodes.tags-> 'addr:street' as street,

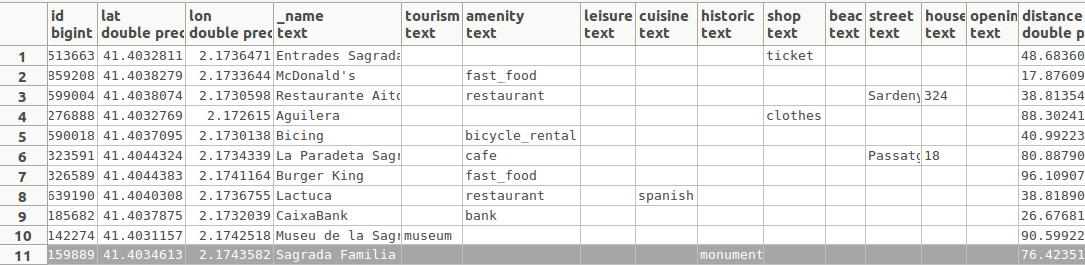
nodes.tags->'addr:housenumber' as housenumber,

nodes.tags-> 'opening\_hours' as openingHours,

ST\_Distance(geography(geom), ST\_SetSRID(geography(ST\_Point(***longitude, latitude***), 4326)) as distance

from nodes

WHERE ST\_DWithin(geography(nodes.geom), ST\_SetSRID(geography(ST\_Point(***longitude, latitude***)), 4326), ***radius***) and nodes.tags ? 'name' and nodes.tags ?| ARRAY['tourism','amenity','leisure','cuisine','beach','historic','shop']



Besides returning the point’s location and organizational information, certain tag keys as presented separately as columns. Based on this information, the application’s *PointConverter* maps the tags to the recommendation categories sightseeing, culture, food, shopping, nature and nightlife (see more in Annex 3, Design Specification).

Bibliography:

1: https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca#.lcnq9f3ro

2: ELIZA – a computer program for the study of natural language communication between man and machine

3: <http://venturebeat.com/2016/08/15/a-short-history-of-chatbots-and-artificial-intelligence/>

4: <https://stanfy.com/blog/advanced-natural-language-processing-tools-for-bot-makers/>

3110: (<https://ronan.collobert.com/pub/matos/2008_nlp_icml.pdf>

3111: (<https://chatbotsmagazine.com/these-five-platforms-will-make-your-bots-language-intelligent-634556750abd>)

3112: <https://docs.api.ai/docs/key-concepts> (API.AI: NLU and Dialog Management)

3113: (Conversational Interface for Smart Devices)

3114: <https://www.fastcodesign.com/3058546/conversational-interfaces-explained>).

3121: <https://docs.api.ai/docs/concept-agents>

3122: <https://docs.api.ai/docs/machine-learning>

3123: https://docs.api.ai/docs/concept-entities

3124: https://docs.api.ai/docs/concept-intents

3125: <https://docs.api.ai/docs/concept-contexts>

320: Charu C. Aggarwal, Recommender Systems, Introduction (p.1)

321: Mining of Massive Datasets

6: <http://wiki.openstreetmap.org/wiki/Points_of_interest>

7: Spatial Databases with application to GIS, **Philippe Rigaux, Michel Scholl**

8: An Introduction to Spatial Database Systems, Ralf Hartmut Güting

9: <http://wiki.openstreetmap.org/wiki/Elements>