UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Development of a Chatbot for Tourist Recommendations

Presentado por Jasmin Wellnitz
en Universidad de Burgos — June 12, 2017
Tutor: Bruno Baruque Zanón

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. Bruno Baruque Zanón, profesor del departamento de Ingenería Civil, Lenguajes y .

Expone:

Que la alumna D. Jasmin Wellnitz, con DNI L28PGM5NZ, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Development of a Chatbot for Tourist Recommendations.

Y que dicho trabajo ha sido realizado por la alumna bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, June 12, 2017

Vº. Bº. del Tutor:

D. nombre tutor

Resumen

Este proyecto consiste en desarollar una aplicación de bote conversacional que es capaz de recomendar puntos de interés turisticos, personalizados a los gustos del usario. Se accede al bote conversacional usando la aplicación de mensajería instantánea Telegram.

El proyecto se focaliza en desarollar el lado servidor del interfaz de Telegram. Para ello se construye un servidor web Java que usa la plataforma de procesamiento de lenguaje natural api.ai para analizar la entrada del usario. Luego la aplicación está desplegada en la plataforma-como-servicio Heroku.

Las recomendaciones dadas al usario están basadas en las informaciones el usario comparte con el bote conversacional y datos ya existentes de otros usarios parecidos. El algoritmo de recomendación aplicado combina los dos enfoques de la teoría de recomendación más importantes: el filtrado colaborativo y basado en el contenido.

Además, se construye un base de datos geográfico usando PostGis, proporcionando las informaciones turisticas en las cuales las recomendaciones están basadas. En la versión de la aplicación presentada, se usa la información geográfica de Barcelona, España y por tal motivo, las recomendaciones proporcionadas están limitadas a esa ciudad.

Descriptores

bot conversacional, interfaz conversacional, Telegram, procesamiento de lenguaje natural, sistema de recomendación, base de datos geográfico, punto de interés...

Abstract

This project deals with the development of a chatbot application that is able to recommend tourist points of interest customized to the user's preferences. The chatbot can be accessed using the instant messaging app Telegram.

The project's main focus lies on the development of a server-side architecture to the Telegram interface. In order to do so, a Java web server is set up that uses the natural language processing platform api.ai to parse the user input. The application is deployed to the platform-as-a-service Heroku.

The recommendations given to the user are based on the information the user shares with the chatbot and already existing data of similar users. The applied recommendation algorithm combines the two most important approaches of recommendation theory, content-based and collaborative filtering.

Furthermore, a geographic database is set up using PostGis, providing the needed tourist information the recommendations are based on. In the presented version of the application, geographic information of Barcelona, Spain, is used and therefore limiting the provided recommendations to this city.

Keywords

Chatbot, conversational interface, Telegram, natural language processing, recommender system, geographic database, points of interest

Contents

List of Figures

List of Tables

Introduction

A chatbot is a computer program that interacts with its users through a conversational interface. They have been a topic of interest in the field of computer science for decades, yet, due to the rise of smart devices in the last couple of years, chatbots have received a great deal of new attention. The possibilities provided by the conversational interface were rediscovered and combined with the state of the art instant messaging methods.

This project concentrates on this upcoming trend by developing a chatbot that can be accessed through an instant messaging platform, in this case the messenger Telegram. Using the messenger's interface, the user interacts with the chatbot and is provided with tourist recommendations. The developed application consists of three principal components:

1. The chatbot which forwards the user input to the natural language processing platform api.ai and interprets the response.
2. The recommendation system which computes the personalized recommendation based on the user's preferences.
3. The geographic information database which makes the needed information for recommendation available.

In the following paper, the developed application is presented by examining the main objectives, underlying theories and way of proceeding as well as describing encountered challenges in the course of the project.

Project Objectives

This project aims to develop a chatbot that is able to present customized tourist recommendations to its users via an instant messaging application. In the presented version of the application, the messenger Telegram is used to interact with the user. The following steps and objectives must be realized to develop the chatbot:

- A Java web server is set up which is connected to Telegram through its webhook API to enable user interaction. The application is deployed to a platform-as-a-service cloud server to enable the webhook integration.
- Design of a conversational interface: the conversation flow between user and chatbot is mapped to the natural language processing platform api.ai which then parses the user input into formalized data. api.ai is accessed by the application through a REST-like HTTP endpoint. The parsed input is interpreted by the chatbot and triggers the desired behaviour, such as recommendation or storage of important user information.
- A recommender system must be implemented to provide personalized tourist recommendations. The recommender is based on data the user has shared with the chatbot and additional data of similar users. To overcome the problem of initially sparse user data, different recommendation methods are combined as well as retrieving existing user data from other sources and/or generating data.
- Retrieval of tourist data from the geographic information database OpenStreetMaps: the data is filtered so that only data of touristic importance is evaluated by the recommender and presented to the user. It is prepared in a way that similarity measures can be made between user interests and the point of interest inherent properties.

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, or more precisely the chatbot, the recommender as well as the geographic database.

3.1 Chatbots

A chatbot is a computer program that interacts with its user via a chat interface. 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 [26], 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.

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 [10]. A sub-discipline of Natural Language Processing is the so-called Natural Language Understanding (NLU).

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 contexts [6].

Using a **conversational interface**, people are enabled to interact with virtual assistants, smart devices and social robot via their natural language [14]. Conversational interfaces are considered as the third wave of user experience, after the terminal and the graphical user interface. The aim is that by having a conversational interface, users 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 [9]. Basically, in chatbots pattern matching is used to interpret the user's inputs and templates are used to provide the system's output.

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 interface.

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 interface [2].

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 conversations [7].

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 [4].

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 [5].

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 [3].

3.2 Recommender Systems

With the advent of business in the Internet, so-called Recommender Systems were introduced and gained importance since the nineties [1]. 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* to the users based on their travel interests. In the following, the applied methods are introduced.

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 based on

OpenStreetMap tags. 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 [22].

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.

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 detail in chapter 4.3.

3.3 Geographic Information Processing

Point of Interest

In the context of geographic information, the expression Point of Interest [19] 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 [23]. This is why for this task spatial database systems are used which provide geospatial data types in its data model and query language [12]. 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*.

OpenStreetMap Data Model

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 [18]:

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.

Principally, the data structures *Nodes*, *Ways* and *Tags* are investigated in detail to construct the wanted *Points of Interests*. The exact proceeding of extracting the needed information from this data model is explained in depth in chapter 4.3.

Tools and Technologies

In this section, the tools and technologies used during the course of the projects are presented. The different components of the geographic database are described as well as the tools to design the conversational interface. Additionally, some alternatives to the eventually used tools are discussed in order to explain why the specific tools are chosen. In contrast, the tools used in the development process such as version control repository, integrated development environment or build-management tools are not explained in detail as they could be interchanged and are not essential for the application's characteristics.

4.1 Chatbot Platform

As already discussed in the previous chapter, there are several competing platforms that permit developers to build their own chatbot. Most of the major software big-players nowadays, such as Google, Facebook, Microsoft, are taking part in the development of these platforms and provide their own solutions, for example API.ai (Google), wit.ai (Facebook), luis.ai (Microsoft) or IBM Watson. Out of these contestants, api.ai and wit.ai seem to be the most widely known and easiest to use. Therefore, both options were investigated to see which one would be the most suitable for this project.

API.ai

API.ai [8] is a natural language processing platform by Google that facilitates creating conversational user interfaces. In order to model conversations, entities and intents are used as key concepts. api.ai provides a rich management toolset as well as a simply one-click-integration mechanism to import the chatbot into a variety of mobile apps. Also, the modeled conversation flow in API.ai can be accessed by submitting queries through the REST-like HTTP endpoints.

API.ai offers a variety of prebuilt agents that can be used as a base for the own conversational interface. On top of it, the conversational interface can be enabled to support *small talk*, allowing it to reply to basic, unspecific user input without any further development.

Wit.ai

Like API.ai, Wit.ai is a natural language processing platform. It was acquired by Facebook in January 2015 and is free to use ever since then [27]. Its key concepts for language processing include entities, intents and, additionally, stories. Stories help to create basic user scenarios in which typical user and chatbot conversations are designed. Regarding integration, Wit.ai provides a web service API to integrate the designed conversational interface in messaging apps.

Choice of Platform: API.ai

After testing both platforms, both left a good impression and seem to be suitable to use. On the one hand, Wit.ai's story feature seems promising to design conversations in an easy way, but is still in beta status. However, API.ai convinces with its rich management toolset, an extensive documentation and the integration of simple conversational features such as *small talk*.

4.2 Geographic Database

In order to extract tourist points of interests from *OpenStreetMap*, a geographic database was set up as the application's backend. The following tools form either a part of the mentioned database or were used to set it up.

OpenStreetMap

OpenStreetMap [17] (or short OSM) is a collaborative project with the aim to collect and update free to use geographic data. Its main purpose is to be a central data source which can be e.g. used for rendering maps. The stored data contains infrastructural information such as roads or buildings as well as variety of additional informational tags. In this project, the OpenStreetMap data is used to extract necessary tourist information in order to create user recommendations. There is a big number of data dumps available that store the above mentioned data for either the whole planet or smaller regions or cities. These dumps can be downloaded in the file formats XML and PBF and imported into a PostgreSQL database.

PostgreSQL 9.5.5

PostgresSQL [21] is an object-relational database system. It is an open source software that is most extensively conform to the SQL standard ANSI-SQL 2008. In this project, PostgreSQL is used to store and manage the geographic database.

pgAdmin III 1.22.0

pgAdmin III [20] is an open source tool to facilitate the management of PostgreSQL databases by providing a graphic user interface. Among many features, it comes with an SQL editor tool to create and run queries and displays data entries.

Osmosis 0.44.1

Osmosis [16] is an open source command-line based application that is able to process data from OpenStreetMap. In this project, it was principally used to import data from OSM files into the PostgreSQL database.

At first, the similar tool *osm2psql* was investigated and used in this project. However, during research it turned out that osm2psql is not the right fit because its main objective is the import of .osm files for rendering purposes. Due to this reason, only data that are render relevant are imported into the PostgresSQL database. On the other hand, Osmosis imports all of the raw .osm data, namely Nodes, Ways, Relations and their corresponding tags [15]

4.3 Messenger - Telegram

Telegram [25] is an instant messaging app for smartphones, tablets or computers. There are available versions for iOS, Android, Windows Phone, as well as desktop applications for Windows, OSX and Linux. Telegram concentrates on speed and security of its messages.

In June 2015, Telegram introduced its Bot API, allowing third-party developers to integrate their own bots into the messenger. The bots are controlled sending HTTPS request. In this project, incoming updates from Telegram are received via an outgoing webhook [24].

4.4 Web Service - Heroku

Heroku [13] is a platform-as-a-service that enables users to run their applications in the cloud. It supports several programming languages, among them Node, Ruby and Java. There are several pricing models offering a range of

different features. In this case, the free plan is used which comes with the inconvenience that the application sleeps after 30 minutes of inactivity, leading to a short delay every time the chatbot is accessed after not using it.

Likewise, Heroku Postgres is a database-as-a-service enabling the upload of the project's database into the cloud.

4.5 Additional Services

Foursquare

Foursquare [11] is a mobile service application which provides personalized recommendations of places for its users. In this project, Foursquare's RESTful API is used to access additional information OpenStreetMap does not provide, more precisely extracting photos for points of interests

Relevant Aspects of Project Development

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (ED-VHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Related Work

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusion and Future Work Lines

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliography

[1] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer, 2016.

[2] api.ai. Agents, 2017. [Internet; accessed 10/06/17].

[3] api.ai. Contexts, 2017. [Internet; accessed 10/06/17].

[4] api.ai. Entities, 2017. [Internet; accessed 10/06/17].

[5] api.ai. Intents, 2017. [Internet; accessed 10/06/17].

[6] api.ai. Introduction - understand the key concepts, 2017. [Internet; accessed 10/06/17].

[7] api.ai. Machine learning, 2017. [Internet; accessed 10/06/17].

[8] api.ai. Welcome, 2017. [Internet; accessed 10/06/17].

[9] John Brownlee. Conversational interfaces, explained, 2016. [Internet; accessed 10/06/17].

[10] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. Technical report, NEC Labs America, 2008.

[11] Foursquare. Detailed docs - api endpoints, 2017. [Internet; accessed 12/06/17].

[12] Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB Journal*, 3:357–399, 1994.

[13] Heroku. The heroku platform, 2017. [Internet; accessed 12/06/17].

[14] Michael McTear, Zoraida Callejas, and David Griol. *The Conversational Interface: Talking to Smart Devices*. Springer, 2016.

[15] OpenStreetMap. Osmosis, osm2postgresql & osm2pgsql – openstreetmap-daten, datenbanken und spielplätze, 2012. [Internet; accessed 10/06/17].

[16] OpenStreetMap. Osmosis, 2016. [Internet; accessed 10/06/17].

[17] OpenStreetMap, 2017. [Internet; accessed 10/06/17].

[18] OpenStreetMap. Elements, 2017. [Internet; accessed 10/06/17].

[19] OpenStreetMap. Point of interest, 2017. [Internet; accessed 10/06/17].

[20] pgAdmin. pgadmin, 2017. [Internet; accessed 10/06/17].

[21] PostgreSQL. About, 2017. [Internet; accessed 10/06/17].

[22] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[23] Philippe Rigaux, Michel Scholl, and Agnes Voisard. *Spatial databases - with applications to GIS*. Morgan Kaufmann Publishers, 2002.

[24] Telegram. Getting updates, 2017. [Internet; accessed 10/06/17].

[25] Telegram. Telegram faq, 2017. [Internet; accessed 10/06/17].

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

[27] wit.ai. Wit.ai is joining facebook, 2015. [Internet; accessed 10/06/17].