Karlsruhe Institute of Technology

Modeling a Dialog System for Movie Recommendations Based on The Movie Database

Bachelor's Thesis of

Kalina-Seslava Nenova

at the Department of Informatics
Institute for Anthropomatics and Robotics (IAR)

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Dr. Sebastian Stüker
Advisor:	M.A. Maria Schmidt

13. January 2016 – 12. May 2016

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung gute wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Karlsruhe, 12.05.2016

. . . .

(Kalina-Seslava Nenova)

Abstract

Many people look for a recommendation, when they are choosing which movie to watch. Some ask friends, others search the internet. In this thesis we introduce another way to get a movie recommendation - using a dialog system, that gets to know the users by learning their preferences and then recommend them movie, they might like.

The goal of this thesis work is to model and develop a dialog system for movie recommendations based on The Movie Database. The dialog system is implemented as text-based and social recommendation system, using a combination of state-based and frame-based dialog control strategies. The result of the system is a system-directed dialog, in which its main aim is a movie recommendation to the user. To accomplish that the system builds and uses a model of the user and a dialog history.

The first part of the thesis gives an introduction in the theoretical fundamentals, which are used in the following development of the system of the thesis, such as structure and types of dialog systems and recommendation systems. The system implementation is described in the second part of the thesis. In the last part of the thesis the user study, conducted to evaluate the system, and its results are outlined.

Zusammenfassung

Viele Menschen suchen nach einer Empfehlung, wenn sie entscheiden, welchen Film zu schauen. Manche fragen Freunde, andere suchen im Internet. In dieser Arbeit stellen wir einen anderen Weg vor, um eine Filmempfehlung zu bekommen - ein Dialogsystem, das Information über die Nutzer und ihre Favoriten bekommt und ihnen dann einen Film empfehlt, der ihnen mit höherer Wahrscheinlichkeit gefällt.

Das Ziel dieser Bachelor Arbeit ist, ein Dialogsystem für Filmempfehlungen auf Basis von der Movie Database zu modellieren und entwickeln. Das Dialogsystem wird implementiert als ein textbasiertes und soziales Dialogsystem mit einer Kombination von zustandsbasierten und rahmenbasierten Dialogsteuerung Strategien. Das Ergebnis des Systems ist ein system-gerichtetes Dialog, dessen Hauptzweck ist den Benutzern einen Film zu empfehlen. Um das zu erreichen, erstellt und verwendet das System ein Modell des Benutzers und ein Dialog History

Der erste Teil der Arbeit gibt eine Einführung in die theoretischen Grundlagen, die in der folgenden Entwicklung des Systems verwendet werden, wie Struktur und Arten von Dialogsystemen und Empfehlungssystemen. Die Implementierung des Systems wird im zweiten Teil der Arbeit beschrieben. Im letzten Teil der Arbeit werden der Benutzer Studie, die durchgeführt wird, um das System zu bewerten, und ihre Ergebnisse geschildert.

Contents

List of Figures

1. Introduction

Humans are social beings. We always interact with the surrounding world, this including not only other humans, but also with the technology. Thus it should be developed according to the human-centered perspective. With the creation of new technology, the human-machine interaction (HMI) is becoming more natural and user-friendly. Years ago this interaction happened by punch card data input with table sized key punches. Looking from our perspective now in 2016, this seems almost impossible to use. Now the input devices are much more easily and naturally operated. The output, that we get from the computer, is also much more convenient and easy to understand. Nowadays, the common way to provide input to the computer is to be using the mouse, keyboard and even touchscreen devices. A big chance for the future is the most natural way by using just our speech.

The voice input and output is a present research problem which is among others tackled in the fields of Automatic Speech Recognition (ASR) and Text-To-Speech Synthesis (TTS). By applying these, HMI systems will get much more natural and easy to use, even when the user is completing other tasks. This way machines will become an even bigger part of our lives. They may become even our new best friend, with whom we may exchange our thoughts, beliefs, wishes, problems etc. We expect a human like answer or a solution to our problems, a recommendation for something we will like. In order to listen to this answer, the machine output should be as natural as possible. To reach this, we will create a human-like natural dialog with the computer. Spoken Dialogue Systems (SDSs) are thus an important factor in the field of HMI. For instance these systems can be used to book a flight, get information about the weather etc.

Recommendation is part of the everyday conversation between humans. The opinion of people we respect effect our own decisions. People ask others, who has similar preferences to theirs, to recommend them something, i.e. movie, restaurant etc, because they expect that they like these. A system can also make recommendation using either this similarities or just recommending users things, similar to these, they like.

The aim of this thesis work is to create a recommending Dialog System for movie suggestions. In the next chapter the theoretical fundamentals of dialog and recommendation systems will be described. In Chapter 3 we will represent the implementation of the created system and in the further chapter - its evaluation. Chapter 5 concludes this thesis and lays out possible future development.

2. Background

In this chapter basic definitions will be outlined, which we will use later in the thesis. First the components of a (Spoken) Dialog System will be described. Then we will explain the difference between text-based (TBDS) and spoken (SDS) dialog system and between goal-oriented and social DSs. At last we will outline different types of roles' distribution and dialog control. In the second part of this chapter we will give a definition to recommendation systems and at the end of the chapter the database we are working with for the implementation will be described.

2.1. Dialog Systems

People engage in dialogs every day with each other. There are many different definitions of a dialog. In this thesis, dialog is defined as a turn taking message exchange between two actors: the SDS and the user. The user may start a dialog with the system for different reasons: to obtain information, to issue instructions, to get a service etc.

2.1.1. Components of a (Spoken) Dialog System

From getting the input from the user to giving them an answer (output) there are five components, that every SDS includes: (automatic) speech recognition (ASR), natural language understanding (NLU), dialog manager (DM), natural language generation (NLG) and text-to-speech synthesis (TTS).

The main input and output of the system is the information from and to the user by means of a speech signal. Between every two components there is a so-called message passing. The output from one module is the input for the next one and so on. This can also be seen on Figure 2.1, which will be explained in the following.

The ASR and the TTS are the modules that differentiates the SDSs from TBDSs. Further comparison between these two types of systems will be drawn in Chapter 2.1.2. The main function of the ASR is to convert the user's speech into text in order to pass it to the NLU and at the end of the system pipeline to the TTS module's task is to transform the text output from the NLG module into naturally sounding speech. In [14] the TTS is compared with the human vocal tract and the ASR with the human ear. The authors of the article also claim that the most successful ASR and TTS are build using the Artificial Intelligence approach and that ASR is the more difficult of the two tasks. However, the ASR and TTS modules are beyond the scope of this thesis.

The ASR module's output is not only one string, but commonly an n-best list of hypotheses. The job of the NLU component is to assign a semantic interpretation to all of these hypotheses. The NLU includes two processes: syntactic and semantic analysis. In the first one, the NLU module has to find the structure of the string, while it has to discover the meanings of the different parts in the second one. There are two main problems of the NLU: natural language ambiguity and ill-formed input. In developing NLU systems there is a conflict between correctness and robustness. John Dowding addresses this problem in [3] and proposes a system that can cope with this issue. Gemini uses a fairly conventional grammar to necessitate the recognition. The grammar is enhanced with two rule-based recognition modules: for glueing the fragments and eliminating disfluencies. One of the most used natural language processing toolkits it the much newer system CoreNLP, developed in Stanford University and described in [7]. CoreNLP is pipeline architecture with a set of robust, stable, high quality linguistic analysis components and provides an easy to use Java API.

The next module, the DM defines system actions as reactions to the input from the NLU module, which it has to analyze. There are three main approaches to process this input: rule-based, frame-based and agent-based/statistical (see Table 2.1). Further the DM has to determine which information should be passed on the NLG module. In order to do that, the DM interacts with (external) knowledge sources: the discourse context, the semantic frame and the database. The dialog history and the user model also affect the management. The most general job of the DM is to control the dialog flow according to the dialog initiative and the type of dialog control, implemented by the developer of the system. These will be further discussed in Chapter 2.1.4 of this thesis.

In order to have a natural output to the user, the information chosen from the DM has to be phrased in sentences. This is the main goal of the NLG: to create natural text from the output of the DM module. The process of the NLG can be divided into three main stages: document planning, microplanning and surface realization. [5] During the document planning it has to be decided which content should be included in the messages to the user and how to structure them. Not all the output from the DM module is convenient to be delivered to the user. In the next phase, microplanning, three main tasks are included: lexical selection, aggregation and referring expressions. In the first task the microplanner has to choose the words and constructs to communicate the information passed from the DM. In the second one it has to be determined how much information each of the sentences should contain. In the last task phrases that identify particular entities are chosen. The microplanner's output is the connection between the sentences. Finally, the surface realization converts this into text and passes it to the TTS module. However, the NLG architecture of Reiter and Dale is too complex for this thesis project, so that we will use another approach and we will implement canned sentences to each type of DM's output.

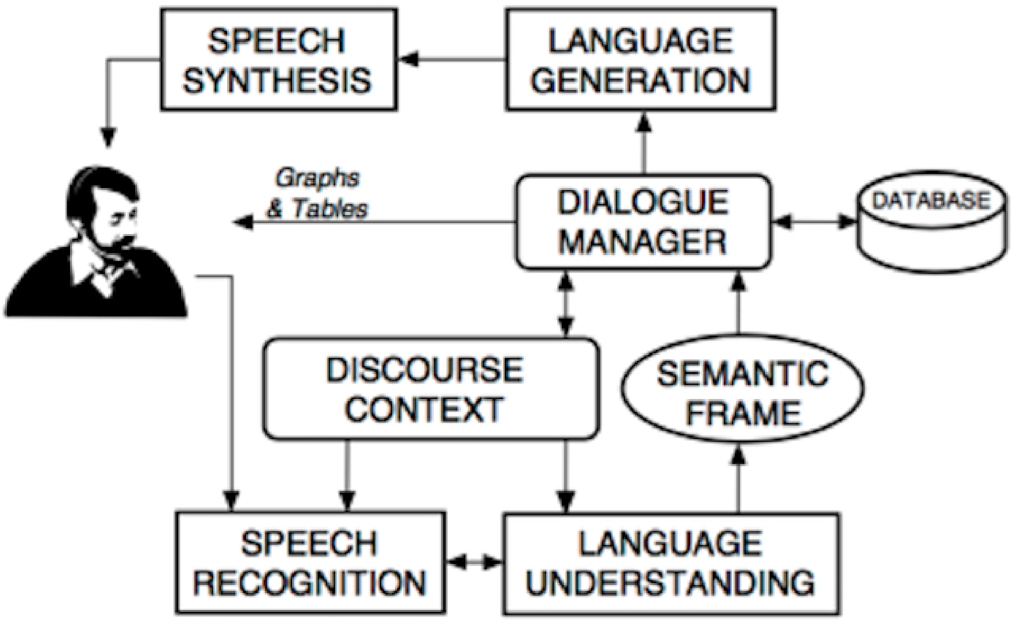


Figure 2.1.: Typical components of an SDS taken from [4]

2.1.2. Dialog Manager Strategies

Dialog Systems can be also differentiated according to how their DM works. There is variation of the dialog initiative, of the dialog control and of the knowledge sources for the DM. Table 2.1 describes the connection between different types of knowledge sources and different dialog control strategies.

Based on that how active the user and the system are taking part in the dialog, there are three types of dialogs: system-directed, user-directed and mixed-initiative. In a system-directed dialog the only active participant is the system. The system asks the user questions in order to get some useful information and get the most suitable data from the knowledge source. The input of the user in system-directed dialog is normally a short answer to these questions. The ASR and NLU modules of such kind of dialogs are easy to implement and less prompt to failure. However, there are less user-friendly, because the user's input is restricted and the user can't take the initiative or change the topic of the dialog. The user-directed dialogs are closer to an interface to a database, in which the queries are coming from the user. The system get's a question from the user and its job is than to look for an answer in the database, give this to the user and then wait for the next question. If the user's input is unclear, the system can ask clarification questions. In order to understand every question from the user, the system has to have much more complex NLU module and the user has to be trained - to know what words and phrases the system can understand. In a mixed-initiative dialog both participants - the system and the user, can have the initiative to ask questions, set topics etc. Such dialog can include shift in the initiative (the user can answer to questions with another question) and gives the user the opportunity to provide more information that was asked in the question. The system has to keep track of all information, given by the user, in order to not ask questions, the user has already answered. In this thesis project we will create system-directed dialog, because the other two types are too complex.

According to [8] there are three types of dialog control: state-based, frame-based and agent-based. As represent in Table 2.1 there are differences between the ways of input and verification, dialog model and user model of each of the system types. There are some similarities between state-based and frame-based system, i.e. they are suitable for form-filling tasks. However, while the order of the states in a state-based system diagram is predefined and inflexible, in frame-based systems the questions do not have to be asked in a fixed in advance flow. Therefore frame-based systems are proper way to implement mixed-initiative dialog systems described earlier. However, frame-based systems are harder to create, because they need to be implemented with "a frame that keeps track of the items of information that the system has to elect from the user; a more extensive recognition grammar and a dialog control algorithm that can determine the system's next actions based on the contents of the frame" [8, page 113]. The most complex among the three types of dialog control is the agent-based one. It is based on techniques from Artificial Intelligence. In agent-based systems the participants in the dialog are seen as intelligent agents, who can understand the actions and wishes of each other. These systems are rather suitable for more type of dialogs, i.e. for discussions and issue solving, than for the form-filling task of state-based and frame-based systems.

Features/Dialog control strategy	State-based	Frame-based	Agent-based
Input	Single words or phrases	Natural language with concept spotting	Unrestricted natural language
Verification	Explicit confirmation - either of each input or at end of transaction	Explicit and implicit confirmation	Grounding
Dialog model	Information state represented implicitly in dialog states. Dialog control represented explicitly with state diagram	Explicit representation of information states. Dialog control represented with control algorithm	Model of system's intentions, goals, and beliefs. Dialog history, context
User model	Simple model of user characteristics or preferences	Simple model of user characteristics or preferences	Model of user's intentions, goals and beliefs

Table 2.1.: Dialog control strategies [taken from [8]]

Typical examples for mixed-initiative architecture and further description about dialog management strategies are mentioned in [11] . The concept of call-flow, around which the simplest finite state control manager (state-based manager) are implemented, is also described in the paper: "call flow is a graph where the nodes represents prompts, and the arcs ... transitions conditioned on the user choice". Popular examples of architecture, mentioned also in the paper, are: AMICA [9] ETUDE [10]. AMICA, created by AT&T labs, is based on the recognition of group of general functions - dialog actions. In its implementation the strategy is designed using recursive transition network with arcs (conditions on the state) and nodes (the dialog function). The created by SpeechWorks International dialog manager ETUDE is a recursive dialog manager, it maintains recursion of the dialog flow. In ETUDE a single node may be extended to a whole dialog.

Different dialog systems need and use different knowledge sources. Ones have to have access to a record of the dialog so far (dialog history), others to the form, to be filled (for instance in mixed-initiative dialogs), also called task record. For social dialog systems for instance the user information, i.e. age, gender and preferences, could also be needed to interact natural with the user. In [8] there are few more knowledge sources described: world knowledge model, domain model and generic model of conversational competence.

2.1.3. Text-Based vs Spoken Dialog Systems

The main difference between TBDS and SDS is, as mentioned earlier, the input from and output to the user. In TBDSs the communication modality is written text and in SDS - speech. The components from TBDS are NLU, DM and NLG, while SDS includes also ASR and TTS. All system components are described in Chapter 2.1.1

The biggest advantage of SDS compared to TBDS is that the users can engage in a dialog while their hands and eyes are busy with other activities. They are also more natural to use and can be used even from blind or analphabetic people. However, they are more prompt to failures, because of natural language ambiguity, and harder for developers to implement. In this thesis project we will create a TBDS.

A typical case of TBDS are chatterbots which aim to interact with the user in such way that user thinks that the program is human. One of the earliest chatterbot is ELIZA [19], created by Joseph Weizenbaum in the 1960s. Although the ELIZA program was implemented with simple pattern matching techniques, it inspired the creation of other important chatterbots. For instance the three times winner of the Loebner Prize [1] , A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) [1] [18] [2], which is one of the strongest chatterbots. A.L.I.C.E. was created by Richard Wallace in the End of 1995. He composed it by using heuristical pattern matching rules. Other popular chatterbots are the Jabberwacky [3] and Cleverbot [4],

[1] www.loebner.net/Prizef/loebner-prize.html

[2]http://alice.pandorabots.com

[3]http://www.jabberwacky.com

[4]http://www.cleverbot.com

both created by Rollo Carpenter with the aim to create an artificial intelligence through human interaction.

2.1.4. Goal-Oriented vs Social Dialog Systems

One can also distinguish the dialog systems on the type of their goals. On one side there are systems, who have a certain goal - goal-oriented systems, and on the other there are systems, whose aim is to create a natural dialog with the user - social dialog systems.

The dialog, composed by goal-oriented systems, has a predefined subject and often cannot be used in situations, which are not connected to this subject. The one, created by a social dialog system, does not have a fixed topic, but their goal is to replicate the conversations, in which human engage everyday.

A popular example of goal-oriented dialog systems are the DARPA Communicator Systems [6] [17] , created in the End of 1990s. The Communicator Systems are 9 systems, designed to support travel planning in form of mixed-initiative interaction. The initial architecture for the project was the Galaxy-II, described in [13] .

The ELIZA system mentioned in Chapter 2.1.2 is an important example of a Social Dialog System and one of the first of its kind. The system uses scripts to interact with the user and the most famous among them is the DOCTOR script, which simulates a psychotherapist. The name Eliza comes from the play Pygmalion by George Bernard Shaw.

2.2. Recommendation Systems

People rely on recommendation for their everyday choices. The WWW depends on recommendation to provide to the users only the information, interesting for them. Recommendation Systems (RS) are the solution for these two problems - they may be used by the user to find a restaurant, a movie, a song, an article etc, that is close to the users preferences; they also may be directly implement in some web sites to show the user only the items, the user might like. For example RS are used in e-commerce web sites, such as Amazon, as part of their product placement strategy.

Based on their knowledge sources [2] separates the recommendation approaches used in RS in four different classes, see Figure 2.2.:

Collaborative: The knowledge source, used by the system, are the rating profiles of different users. Collaborative systems detect peer users, which rating history is similar to the current user's one, and generate recommendations using this similarity. An example for a RS that uses the collaborative approach is the Groundy System, described in [12] . The system uses stereotypes to build models of individual users. Another examples of such systems are the book RS of Amazon and PHOAKS [15] , used by the WWW to recommend a relevant web resources.

Content-based: The system uses two knowledge sources: item's characteristics and the valuation that a user has given them. Recommendation is seen as a user-specific classification problem. To solve this problem a Bayesian classifier is trained to estimate the probability if the item is liked by the user or not. Content-based RSs have various disadvantages, such as limited content analysis, overspecialisation and the new user problem. The last is also a limitation of the Collaborative RSs.

Demographic: A demographic recommender uses the demographic profile of the user, which might include user's age, sex, nationality etc, as a knowledge source for the recommendation. For different demographic niches can be made different recommendations, by combining the ratings of users in those niches.

Knowledge-based: The knowledge source, used by knowledge-based recommenders, is the interpretation of user's needs and preferences. This knowledge could consist of explicit functional knowledge about the probability with which item features meet user needs.

All of the techniques have limitations. This is why hybrid RSs are used to improve the recommendation and solve the cold-start problem. The hybrid RSs are described in [2, page 380] as "any recommender system that combines multiple recommendation techniques together to produce its output". In the paper there are seven different strategies for Hybrid Recommendation described: weighed (numerical combination of the score of different recommendation techniques), switching (execution of the selected by the system technique), mixed (simultaneously recommendation from different recommenders), feature combination (single algorithm using combination of knowledge from different sources), feature augmentation (knowledge computed by one technique and used by the next one), cascade (recommenders with priority) and meta-level (a model produced by one technique and used by the next one).

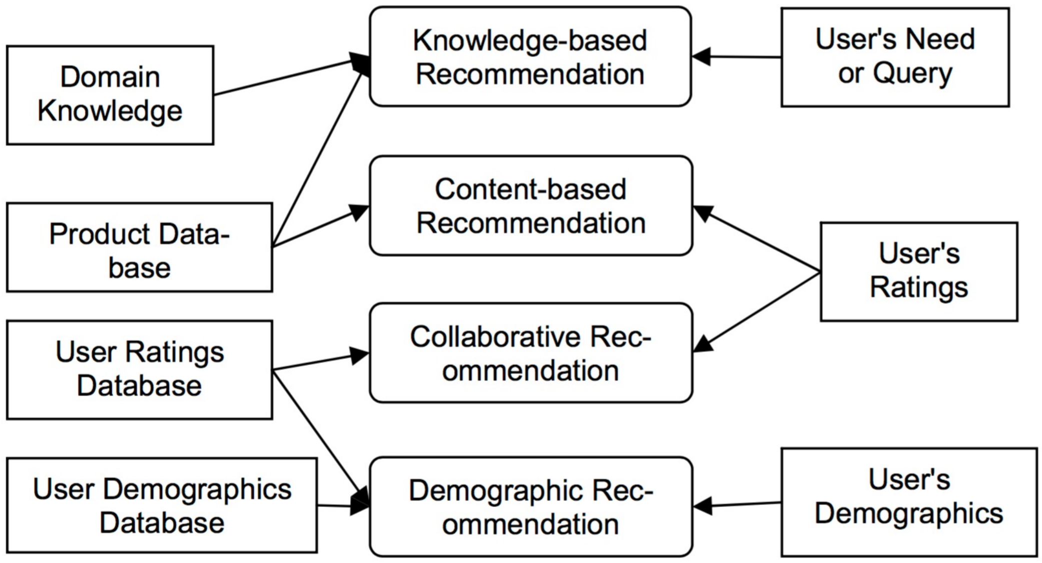

Domain Knowledge
Product Data-base
User Ratings Database
User Demographics Database
Knowledge-based Recommendation
Content-based Recommendation
Collaborative Rec-ommendation
Demographic Rec-ommendation
User's Need or Query
User's Ratings
User's Demographics

Figure 2.2.: Recommendation techniques and their knowledge sources [taken from [2]]

2.3. The Movie Database (themoviedb.org)

For the implementation of the system of this thesis work we used The Movie Database (TMDb) as source of information. In this chapter we will describe the whole platform and in Chapter 3.4 we will outline what we used from the API and how we integrated it in our system.

As described in [16] TMDb is editable website, just like Wikipedia, but with only information about movies, TV and actors. The project started in 2008 and the initial data in form of 10000 movies came from the Open Media Database [5] - similar platform for film media. Since then the editors and contributors to the TMDb are the users and this way it became "one of the most actively user edited movie database on the Internet." Nowadays there are 283,169 movies, 63,687 tv shows with 1,135,479 tv episodes, and 724,816 people on TMDb.

The TMDb API can be used to programmatically get data from the DB and use it in an application. There are several apps for different operating system, already created with the API, such as LaLune[6] for iOS, Movie Tip[7] for Android, Film Closet[8] for Windows and Windows Phone, DVDpedia[9] for OS X and more.

[5]http://www.omdb.org/content/About
[6]https://itunes.apple.com/us/app/lalune-movies-and-tv/id1033048573?mt=8
[7]https://play.google.com/store/apps/details?id=com.okster.movietip
[8]https://www.microsoft.com/en-us/store/apps/film-closet/9wzdncrfjb28
[9]http://www.bruji.com/dvdpedia/

For movies and tv shows the main features of the API are Search, Discover, Find and Get Details. The application developers may use the Search feature to search with a text query, the Discover feature to search based on data like release/air dates or genre, the Find feature to find movies or tv shows based on external ID, and the Get Details feature to get further information about a movie or a tv show. For People there are also Search, Find and Get Details features, which work in similar way. Besides that the API provides more general information, such as list with top rated, popular or upcoming movies and tv shows.

3. Implementation

In this chapter we will describe the system of this thesis work and its implementation. The system creates a user model with users' movie and actor preferences and than either recommend movies to the users or give them some information about their favourite movies or actors.

The first section of the chapter gives an overview of the whole system and its architecture. The separate modules are then described in the remaining sections of the chapter. The modules are divided into two parts: system modules and knowledge modules, based on the task they fulfil. In the last section we outline the domain, used in the system, namely the movie database (DB) [1]. We also explain the usage of the API, with which the connection to the DB was established.

3.1. Overview

In order to easily make changes to the system, we designed its architecture features modular. (see Figure 3.1) This way the flow of the system and the implemented algorithms are more extendable and simple to adjust. As mentioned above we differentiate between system modules and knowledge modules. The system modules are the components of the TBDS, see Chapter 2.1.1 for a general explanation of the components of SDS and TBDS, whereas the knowledge modules represent the knowledge sources, used by the dialog manager, and their operation.

3.2. System Modules

The system, implemented in the scope of this thesis work is TBDS, therefore we implemented only the modules of such kind of systems: NLU, DM and NLG, described in Chapter 2.1.1 The NLU module and the NLG module are implemented as simple as possible. The NLU module extracts information from the user input and passes it to the DM. The DM then passes the needed information for the output and passes it to the NLG module.

[1]themoviedb.org

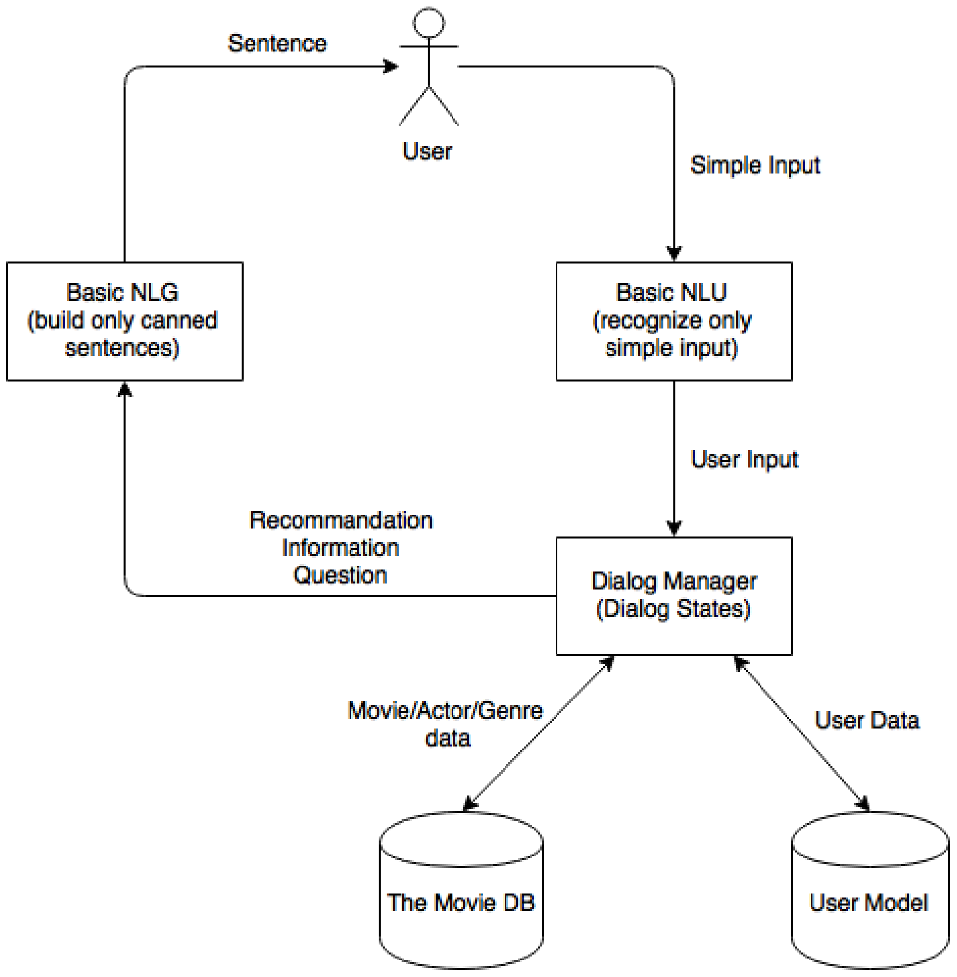


Figure 3.1.: System Architecture

3.2.1. Language Understanding

The NLU module is implemented by defining regular expressions for every type of user input in every state. Using these regular expressions the sentences, typed in from the user, are clean and from them only the important information is passed to the system. For instance, if the user is asked for favourite actor or movie, the system deletes input such as "My favourite ", "actor"/"movie", "is", "I like" etc. from the sentence and gets only the name/title of the actor/movie.

3.2.2. Dialog Manager

The dialog, which the implemented system creates with the user, is system-directed one. All of the questions are coming from the system and the user can answer them only with simple input and can't give any additional information.

For the implementation of the dialog flow we used a combination of state-based and frame-based dialog control strategy. The dialog flow contains states, such as information giving state, recommendation state, asking state etc., but their order is not predefined. The dialog flow diagram is presented in Figure 3.2. The main flow states of the system are: Start, Ask, Recommend and Info. The system flow begins always with the Start state, continuous into the Ask state to start the conversation about movies and then goes into the Recommend or Info state, depending on the dialog history. With the help of dialog history every state comes after another state in order to for example not have two questions following after each other. In Chapter 3.3.2 the implementation and usage of the dialog history is further explained.

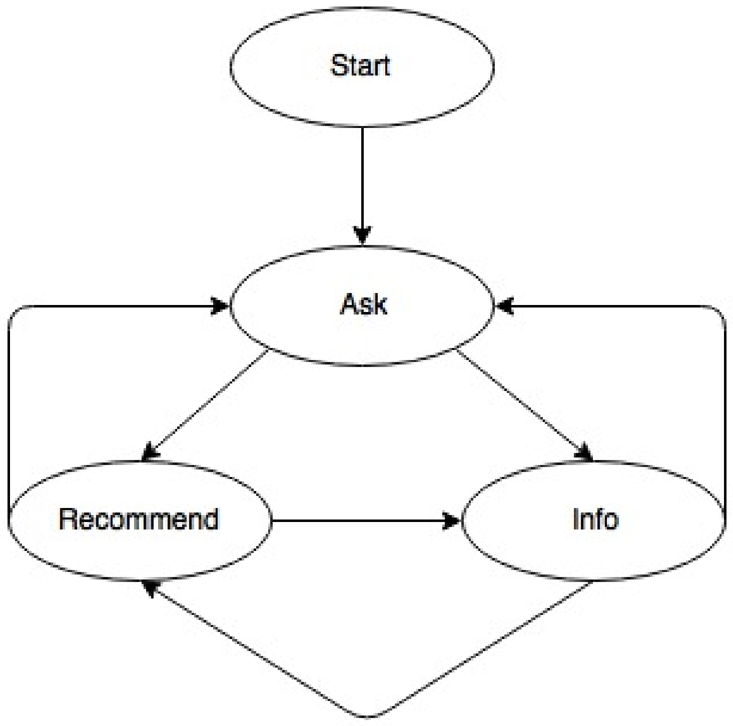


Figure 3.2.: Flow diagram of the dialog states

At the beginning of a dialog, in the Start state, after greeting the user the system first asks the user for username and checks if the username is unique. If not, this could mean two things: the user is old or the user is new, but the selected username is already used by another user. If so the user is asked to give a new username until the username is unique. At that point the system creates new user entity with the username and further asks the user about the user's age and name to add them to the user profile. In Chapter 3.3.1 further explanation about the user model and its purpose is given. The flow diagram of the Start state can be seen in Figure 3.3 and the code of the Start state implementation can be seen in Chapter A.1.

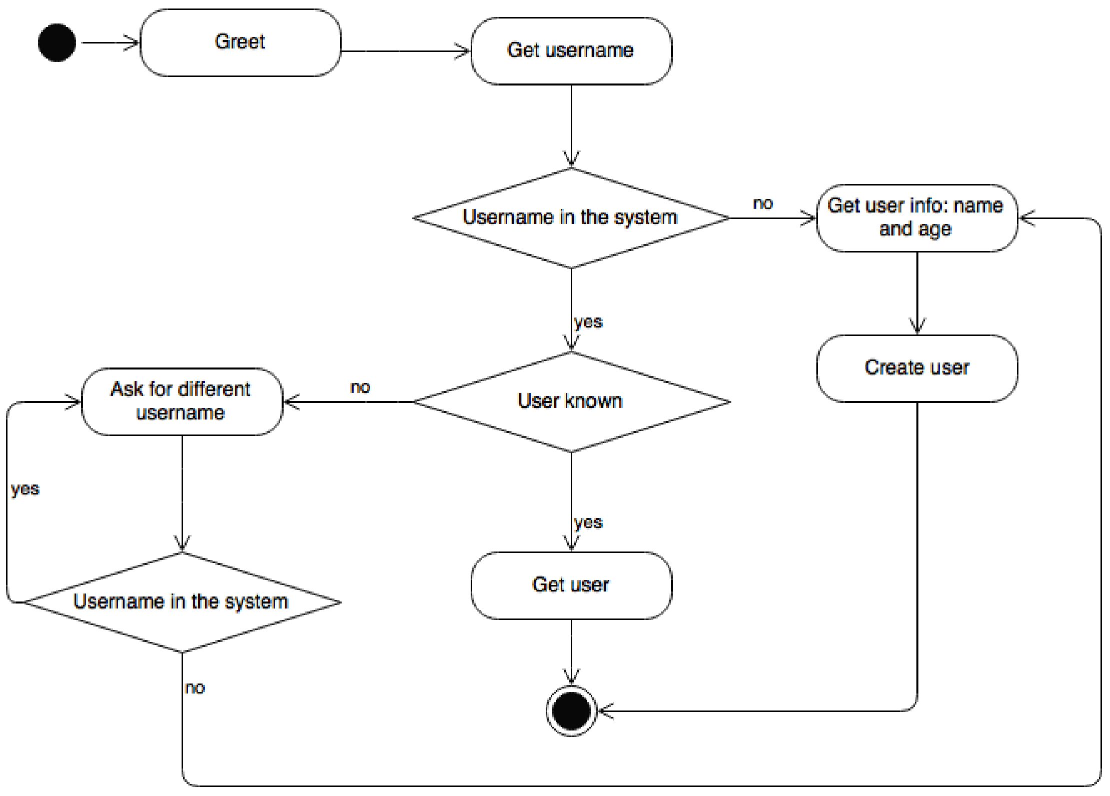
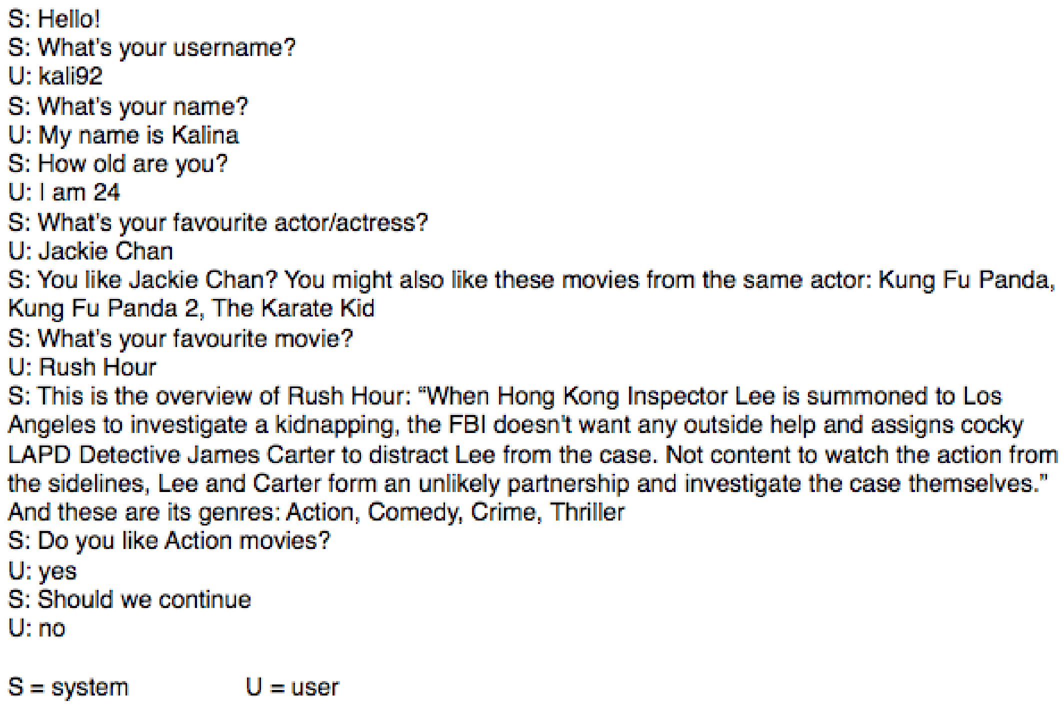


Figure 3.3.: Activity diagram of the Start state

The initial input to the system and the input to the Start state is, as mentioned above, the username. The output of the Start state is a user object. Except for the Start and the Ask state, the input and output of every state is a user object, containing the changes, made in the state. For instance the Recommend state takes the list of already recommended movies from the user object in order to not recommend twice the same movie, makes the recommendation and puts the recommended movie into the list.

3.2.3. Language Generation

As mentioned in Chapter 2.1.1 we used canned sentences to implement the NLG module of the system. The sentences contain slots, where the input from the DB, such as information about movie or actor, or movie title list, is added. Figure 3.4 shows an example of a dialog between the system and a new user.



S: Hello!
S: What's your username?
U: kali92
S: What's your name?
U: My name is Kalina
S: How old are you?
U: I am 24
S: What's your favourite actor/actress?
U: Jackie Chan
S: You like Jackie Chan? You might also like these movies from the same actor: Kung Fu Panda, Kung Fu Panda 2, The Karate Kid
S: What's your favourite movie?
U: Rush Hour
S: This is the overview of Rush Hour: "When Hong Kong Inspector Lee is summoned to Los Angeles to investigate a kidnapping, the FBI doesn't want any outside help and assigns cocky LAPD Detective James Carter to distract Lee from the case. Not content to watch the action from the sidelines, Lee and Carter form an unlikely partnership and investigate the case themselves."
And these are its genres: Action, Comedy, Crime, Thriller
S: Do you like Action movies?
U: yes
S: Should we continue
U: no

S = system U = user

Figure 3.4.: Example of a dialog

For each dialog state, see Figure 3.2, we created a collection of generated sentences. For instance there is a collection of questions for the Ask state, a collection of recommending sentences for the Recommend state and a collection of informative sentences for the Info state. For the Start state we created a special collection of questions and greeting sentences.

In order for the output of the system to sound more natural, we created a set of sentences, which can replace each other. We use a different sentence from a set in order to not repeat the same one in the whole dialog. This sentence is chosen randomly among the others. For example the greeting sentence can be: "Hello!", "Hi", "Good morning/afternoon/evening/night!" etc. In the last example we select the greeting according to the time of the day.

3.3. Knowledge Modules

The dialog manager of the system uses two different knowledge sources, both created and developed during the dialog. The first one, the user model, represents the information about the user and will be described in Chapter 3.3.1 The second one, the dialog history, used by the system to save information about the dialog flow, will be introduced in Chapter 3.3.2

3.3.1. User Model

The user information is saved in a user object. The class diagram of the User class can be seen in Figure 3.5. The user object is saved with unique username, user age and name, a list with user's favourite movies, favourite genres and favourite actors. In the user object the system also saves the information, already given to the user, in form of a hash map and a list with the movies, which are already recommended to the user. Previous dialog history, as described in Chapter 3.3.2, is also saved in form of list of integers.

All user objects are saved in a single hash map. This provides more dynamical access to a user object, i.e. searching specific user among all users and saving user information to the right user object. This hash map is then saved in a file. This way the hash map and the user objects in it are saved for further use of the system.

When the system is started, the DM gets the hash map from the file, so that it can use it in the whole dialog flow. At the beginning of the interaction with the system, the user is asked for username (see Figure 3.3.). The DM then searches in the hash map for this user. If the user is an old one, the DM gets the specific user object from the hash map, so that it can get and put information to it during the dialog. At the end of the interaction, the user object is put back to the hash map and the hash map is put back into the file.

3.3.2. Dialog History

The dialog history consist of list of strings with the titles of the already run dialog flow state (ask, recommend, info), list of strings with the already discussed topics (movie, actor, genre), integers for all of the things, that might have been asked (movie, actor, genre, year), containing how many times the specific thing has been asked.

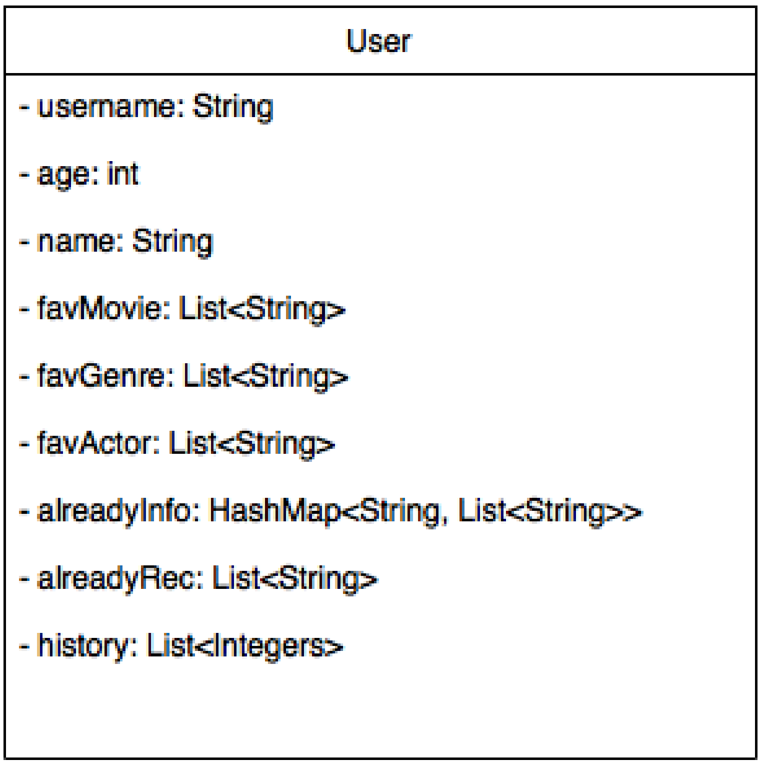


Figure 3.5.: Class Diagram of the User Object

The list of state names is used by the DM to check what is the latest run state in order to not repeat states, especially to not repeat the Ask state. After each state, its name is added to the end of the list. The list of discussed topics is used in analog way.

The integers are used in order for the DM to figure out, which topic has been discussed at least. The DM calculates the minimum of all the integer values and sets the next topic to the topic with this integer. After a new topic is discussed, its integer is increased by one. At the beginning of interaction between the user and the system, the DM gets these integers from the user object, if it is not new, or sets them to 0, if the user is new. At the end of the dialog, the DM saves the integers back to the User Object.

3.4. The Connection to The Movie DB

As described in Chapter 2.3 one can use the API of TMDb to fetch data from the database and then use it in the implementation of an application. There are two main ways to do that. On one side the developer can use direct the API and call its functions and on the other the connection to the API can be established using wrappers and libraries, created by other users of TMDb. We choose to follow the second approach and use such help.

Because the system of this thesis work is implemented with Java, we decided to use an API, that is also written with and for Java. The API was created by Stuart Boston and for use by YetAnotherMovieJukebox[2], also developed by him. In the API there are classes and functions for all features of TMDb, some of which are described in Chapter 2.3.

At the first stage of the implementation of the system we learned what we can do with the API and how to do it. After that we choose the functions, that are relevant for our system, and divided them in two groups: movie and people methods. For both groups we implemented a method to get the ID of the movie, genre or actor. Further using this ID we get the relevant data from the TMDb such as similar movies and movies from specific genre or actor, to recommend them to the user. We also implemented methods to get the additional information, which should be later given to the users, using the ID. We implemented methods for the general features of TMDb to recommend top rated or popular movies to the user, too.

TMDb is the main source of data for the system, created in the scope of this thesis. It was of big importance to learn how to work with the API of TMDb and also with the wrapper API to establish the connection to TMDb. In further development of the system, one can use more features of TMDb and implement the with the help of these APIs.

[2]https://github.com/YAMJ

4. Evaluation

After the system of this thesis was implemented, a user evaluation study was conducted. We choose such kind of evaluation, because the main actor of the system is the user as participant in the dialog, conducted by the system.

In the fist part of this chapter we will outline the goals of the evaluation. Further, in the next part, we will represent the study design. In the further two parts of this chapter we will report the results of the study and discuss them. In the last part of this chapter, we will summarise the evaluation.

4.1. Goals

The main goal of the system, created in the scope of this thesis, is to engage the user into a dialog on their movie, actor or genre preferences and make some recommendation, using the information, gathered in the dialog. The main goal of the user study is to evaluate if and how well this goal is achieved. This study aims at determining how interesting the recommendation for the user is. The usability and error handling are also examined by the study.

Not only the achievement of the main goal of the system, but also the performance of the separate modules involved in the implementation should be evaluated during the user study. This way even if the system fails it main goal, the possible reasons for this failure can be found and analysed.

In conjunction with these goals, the user study also attempt to establish the opinion of the user on the further usage of the system. The user is also asked to give some feedback and suggestion how the system could be improved to gain bigger user satisfaction.

4.2. Study Design

After the goals of the evaluation had been defined, we started with the design of the user study. We decided to initiate a live test study where the participants could try out the system and then rate its performance in a questionnaire. One of the advantages of this kind of user study is that the users will evaluate a movie recommendations, that are made according to their preferences.

When we decided to initiate a live test study, we needed also to determine how to present the system to the study participants. We agreed on inviting each of the participants to a personal meeting, where they can use the system on the computer, on which it was implemented. After the users have left the dialog with the system, we asked them to fill the questionnaire and give us also some oral feedback. Most of the participants were also computer science students, so they also wanted to see how the system was implemented and give us also feedback on the code.

In order to evaluate how the system works for new and known users, we asked the participants to run it twice. It was important to see if the system uses properly the information, given by the user in the first part, and if it maintain interesting for the users even if they run it many times. The questionnaire was also divided into two main parts, which we will describe later in this chapter.

After it was decided how to conduct the live study, it was time to design the last component of the user study: the questionnaire. Its main purpose was to collect information from the study participants which will be used to fulfil the study's goals, that we described in Chapter 4.1 The questionnaire was divided into four parts: demographic data and general information, satisfaction after the first run of the system, satisfaction after the second run of the system, overall satisfaction.

The first part of the questionnaire, the general information, the participants fill before the first run of the system. There they were first asked for their age and gender. This demographic data is widespread in user studies. Further the participants answered questions, which are relevant for the particular goals of the evaluation. These questions aim to determine the amount of movies the participants watch and the experience they have with other recommendation systems.

Later after the first run of the system the participants were asked questions to determine their satisfaction with the system. They gave feedback on the content of the system's output and also on its structure. For instance, the participants were asked if the recommended movies were interesting for them and if the system output seems natural to them. Further questions from the questionnaire can be found in Chapter A.2.

The part of the questionnaire after the second run aims to measure user satisfaction after the second run. The user is already known to the system so the expected result is, that the system uses at least to some extent the information, given by the user in the first run, and does not repeat recommendations. Some of the questions from the part of the questionnaire after the first run are used again in the part after the second run to determine if the content is still interesting for the user.

In the last part of the questionnaire the user is asked to express opinion about the future usage of the system. The participants answer questions about their overall satisfaction with the system, as well as express their expectations if the system could be used in the future in everyday life. At the end of the questionnaire the participants have the opportunity to give a feedback on the whole system in form of a text. For instance, they answer if the system met their expectations and give suggestions for further improvement.

We create the survey using LimeSurvey [1] in order to make it available online and for further evaluation of the system. The questions in the questionnaire are of different type, but the most we designed as Yes/No questions or we used the 5-point Likert scale.

4.3. Results

In order to examine the results of Likert scale questions, we assigned numerical values to the answers. Positive answers are assigned bigger numbers. In Table 4.1 one can see the exact assignment of the values.

Ten participants took part in the user study. All of them answered the demographic questions. Their gender was equally proportioned: five male and five female participants. The age distribution was in range between 22 and 29 with average and median of 24.

Non of the participants answered that they watch movies every day or more times a week. Three watch movies every week, two more times a month, two every month and three watch movies not every month. Two of the participants use recommender systems very frequently and three - frequently. The rest use such systems either very infrequently or do not use at all. Two participants answered that they use Imdb and one that use YouTube.

[1]https://www.limesurvey.org

1	2	3	4	5
Very boring	Boring	So-so	Interesting	Very interesting
Very artificial	Artificial	Somewhat artificial and natural	Natural	Very natural
Strongly disagree	Disagree	Neutral	Agree	Strongly agree

Table 4.1.: Numerical values of the answers

In terms of how interesting the recommended movies for the participants were, the system was rated after the first run on average with score of 4.2 and after the second run on average with score of 4.1. The content of the additional information, given by the system after the first run, was rated on average to 3.9 and after the second run to 3.8. The distribution of the answers to these questions can be seen in Figure 4.1 and Figure 4.2.

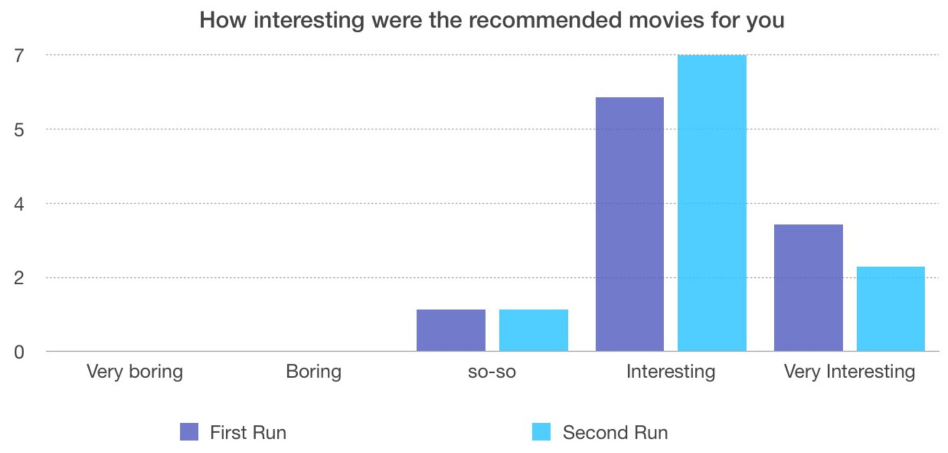


Figure 4.1.: Distribution of the answers of the question "How interesting were the recommended movies for you?" after the first and the second run

After the first run one of the participants answered that the given information was new to them, two learned nothing new and the rest seven received information that was to some extent new. After the second run the information was new for

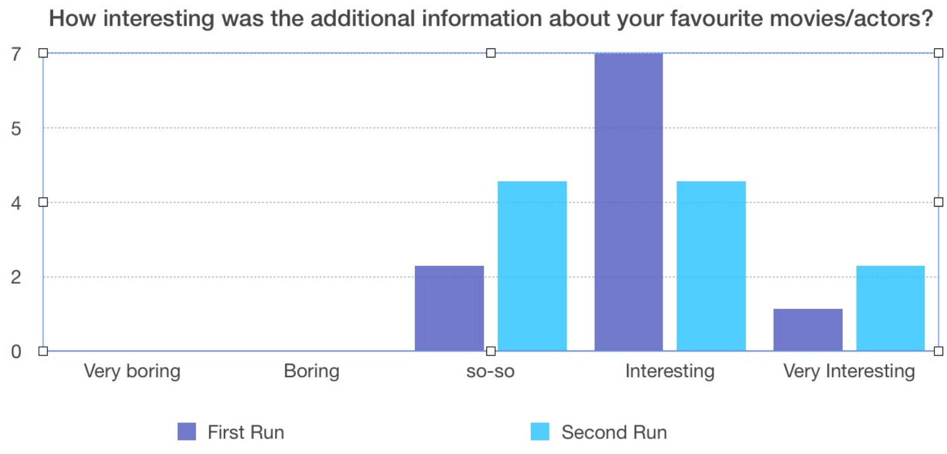


Figure 4.2.: Distribution of the answers of the question "How interesting was the additional information about your favourite movies/actors?" after the first and the second run

three participants, old for one and to some extent new for the rest six. In Figure 4.3 the distribution of these answers is illustrated.

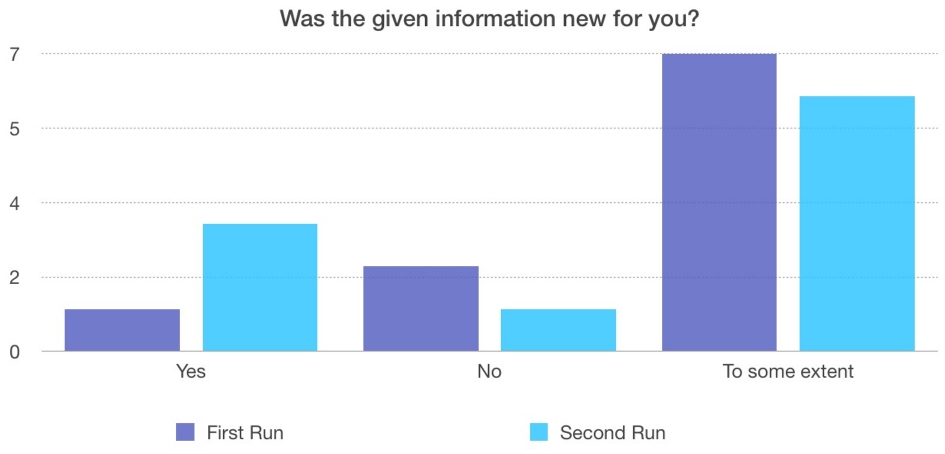


Figure 4.3.: Distribution of the answers of the question "Was the given information new for you?" after the first and the second run

In Figure 4.4 one can see the distribution of the answers of the questions about the usage of information from the first run in the second one. Eight participants reported that the system gave them new information and recommended them new movies and two participants answered to these questions, that the system did that only to some extent. The system used the information from the second run for the dialog in the second run with six participants, with three the system did that only to some extent and one reported, that the system did not use any information from the first run.

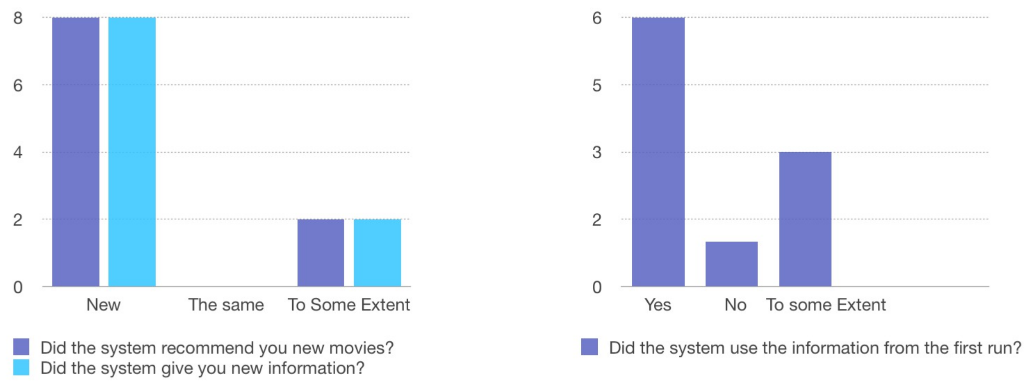


Figure 4.4.: Distribution of the answers of the questions, measuring if the system used properly the information from the first run

On the question about the form of the system output three participants answered, that for them the output was somewhat artificial and natural. Four participants think that the output is natural and to three participants the output seems very natural. Because no one gave to this question the two negative answers, there was not any further explanation about the naturalness of the output.

Figure 4.5 represents the answer distribution for the last part of the questionnaire, the overall satisfaction. The average score for this satisfaction is 4.3. To the question if the participants will continue using the system the average score is 3.7 and to the question if they would recommend the system to friends - 4.1. Most of the participants can imagine people using the system on daily basis.

Only three of the participants gave further feedback in the free input form. They found the idea interesting and the system output good. However, they also point out the weak points in the system. For instance, they think that the interface could be better and offer to the users an opportunity to correct their input. One of the users also mentioned, that the list of movies contains few movies, that he has not watched yet.

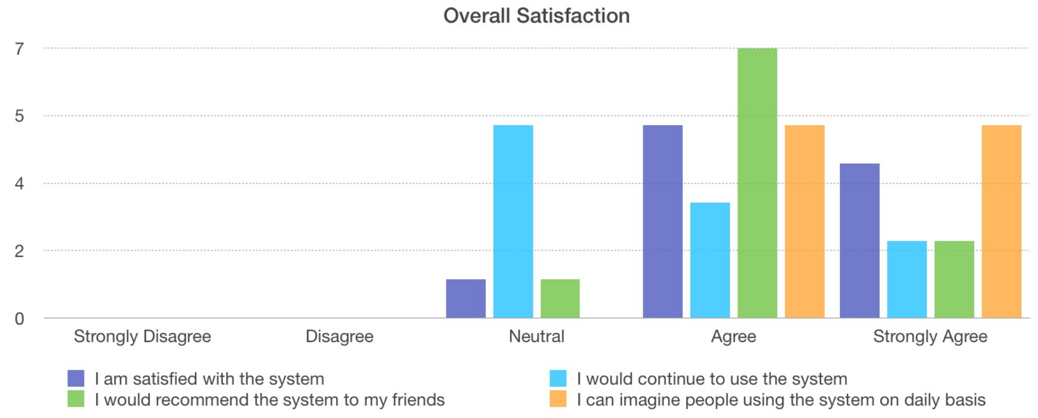


Figure 4.5.: Distribution of the answers of the questions, measuring the overall satisfaction

4.4. Discussion

From the results of the questionnaire we can see that there isn't big difference between how interesting the content was after the first and after the second run. After the second run some participants change their answers with 1 point, but the average change is only 0.1 which means that for the most participants the content is equally interesting after the first and second run.

Interesting observation is that the information after the second run was new to more participants than the one after the first run. A possible reason for that is the fact, that in the first run the users speak about their favourite actors or movies, about which they know more, and then later in the second run the topic is their second favourite ones and so on. Another possible cause of this result may be the structure of the list of recommended movies. The API orders this list and puts the most popular movies on the top of them.

The results of the questionnaire show that the system may not use the dialog history from the first run properly. The dialog history is further explained in Chapter 3.3.1 For instance, two of the participants reported that the system did not recommend them only movies, which were not recommended in the first run. However, this result may be produced by misunderstanding of the question.

The answers of the question about the naturalness of the system output show us that the NLG module produces a natural sentences. This is important for our system, because it should be a social one. However, because of the answers of the participants, to which the output seem only somewhat artificial, we should invest further in implementing an even better NLG.

Although most of the participants are satisfied with the system and it meets their expectations, many did not answered that they would continue to use the system. This may be caused by the fact, that the participants in the survey do not watch movies that often. However, they would recommend the system to their friends. This can be interpreted as will to use the system, if they watch movies and need movie recommendations.

4.5. Summary

The result of the user study demonstrate that the system, implemented in the scope of this thesis work, accomplishes its main goal to produce movie recommendation and give information, which are interesting for the user. Further, we can observe, that the system succeed to produce an interesting dialog even if the user has used the system before. The feedback on the structure of the output shows that the system generate naturally sounding sentences, which is also important, because we aim to create a social recommendation system.

However, the evaluation shows also some downsides of the system, that need improvement. The biggest one is that the user does not have any freedom and opportunity to direct the dialog, correct the given input or make complex input, containing of more then one movie title, genre or actor name. These issues should be addressed in further development of the system.

5. Conclusion

In this thesis we created a dialog system for movie recommendations based on The Movie Database - a web platform with movie data. The system combines the user friendliness of a dialog system with the practicality of a recommendation system. This chapter gives some conclusion to the thesis and introduces some ideas for further development of the system.

We started the work by exploring the theoretical backgrounds of dialog and recommendation systems to decide which approach we should follow. We made the decision to develop a text-based dialog system with combination of state-based and frame-based dialog control strategies. We also implemented a user model and dialog history, that the dialog manager should use in the dialog flow.

After the implementation was ready, a user study was conducted in order to evaluate the system. Ten participants took part in the evaluation process and most of them gave a positive feedback. However, the user study's results show also some gaps in the system, that need to be improved in further development.

One of the most important improvement, that should be made, is that of the dialog manager. The developer can work on changing the dialog control strategy to create a mixed-initiative dialog and gave the users an opportunity to change the dialog flow, correct their answer or make an input with more then one entities (move title, actor name etc.)

An improvement of the dialog manager is tightly connected to further development of the NLG and NLU modules. The NLU module should recognise correct all entities in the users' input. The current implemented NLU with regular expressions cannot do that and it should be redesigned and implemented with the help of some of the systems, described in the background section on NLU. The NLG should also be improved to react more adequately to the user-generated dialog flow and produce more naturally sounding sentences.

Some of the evaluation participants asked if the system works only with movies or also with tv shows. In further expansion of the system the Movie Database can also be used more widely and to its full potential. Not only movie and actor data can be fetched from the platform, but also tv shows and episodes. Thus the system will be improved from only movie to movie and tv shows recommendation system. This way the dialog will get more interesting for the users.

After making the mentioned improvements, the system will become even more user-friendly and practical to use. It might be used by a lot of people on daily basis and might be also embedded in other dialog systems to create a one unified social dialog system, with which the users can have conversation on many different topics, one of which will be their favourite movies and tv shows.

Bibliography

[1] http://www.alicebot.org/anatomy.html

[2] Robin Burke. "Hybrid web recommender systems." The adaptive web. Springer Berlin Heidelberg, 2007. 377-408.

[3] John Dowding et al. "Gemini: A natural language system for spoken-language understanding." Proceedings of the 31st annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1993.

[4] James Glass. "Challenges for spoken dialogue systems." Proceedings of the 1999 IEEE ASRU Workshop. 1999.

[5] Ehud Reiter, Robert Dale and Zhiwei Feng. Building natural language generation systems. Vol. 33. Cambridge: Cambridge university press, 2000.

[6] Esther Levin et al. "The AT&t-DARPA communicator mixed-initiative spoken dialog system." INTERSPEECH. 2000.

[7] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit." ACL (System Demonstrations). 2014.

[8] Michael F. McTear. Spoken dialogue technology: toward the conversational user interface. Springer Science & Business Media, 2004.

[9] Roberto Pieraccini, Esther Levin and Wieland Eckert. "AMICA: the AT&t mixed initiative conversational architecture." Eurospeech. 1997.

[10] Roberto Pieraccini et al. "ETUDE, a recursive dialog manager with embedded user interface patterns." Automatic Speech Recognition and Understanding, 2001 IEEE Workshop on. 2001.

[11] Roberto Pieraccini and Juan Huerta. "Where do we go from here? Research and commercial spoken dialog systems." 6th SIGdial Workshop on Discourse and Dialogue. 2005.

[12] Elaine Rich. "User modeling via stereotypes*." Cognitive science 3.4 (1979): 329-354.

[13] Stephanie Seneff, Raymond Lau and Joseph Polifroni. "Organization, communication, and control in the GALAXY-II conversational system." EUROSPEECH. Vol. 99. 1999.

[14] Douglas O. Shaughnessy. "Interacting with computers by voice: automatic speech recognition and synthesis." Proceedings of the IEEE 91.9 (2003): 1272-1305.

[15] Loren Terveen et al. "PHOAKS: A system for sharing recommendations." Communications of the ACM 40.3 (1997): 59-62.

[16] https://www.themoviedb.org/faq/general

[17] Marilyn A. Walker et al. "DARPA communicator dialog travel planning systems: the june 2000 data collection." INTERSPEECH. 2001.

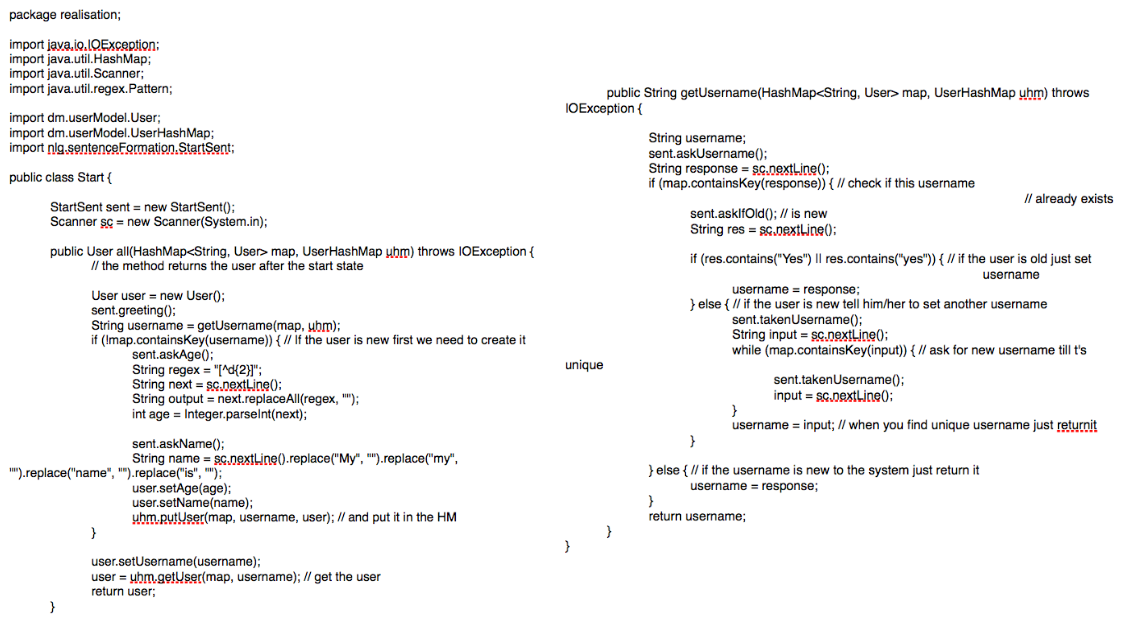
[18] Richard S. Wallace. The anatomy of ALICE. Springer Netherlands, 2009.

[19] Joseph Weizenbaum. "ELIZA - a computer program for the study of natural language communication between man and machine." Communications of the ACM 9.1 (1966): 36-45.

A. Appendix

A.1. Code example

In Figure A.1 the code of the Start state is presented. The flow diagram of the Start state can be seen in Figure 3.2.



```
package realisation;

import java.io.IOException;
import java.util.HashMap;
import java.util.Scanner;
import java.util.regex.Pattern;

import dm.userModel.User;
import dm.userModel.UserHashMap;
import nlg.sentenceFormation.StartSent;

public class Start {

        StartSent sent = new StartSent();
        Scanner sc = new Scanner(System.in);

        public User all(HashMap<String, User> map, UserHashMap uhm) throws IOException {
                // the method returns the user after the start state

                User user = new User();
                sent.greeting();
                String username = getUsername(map, uhm);
                if (!map.containsKey(username)) { // If the user is new first we need to create it
                        sent.askAge();
                        String regex = "[^d{2}]";
                        String next = sc.nextLine();
                        String output = next.replaceAll(regex, "");
                        int age = Integer.parseInt(next);

                        sent.askName();
                        String name = sc.nextLine().replace("My", "").replace("my",
"").replace("name", "").replace("is", "");
                        user.setAge(age);
                        user.setName(name);
                        uhm.putUser(map, username, user); // and put it in the HM
                }

                user.setUsername(username);
                user = uhm.getUser(map, username); // get the user
                return user;
        }

        public String getUsername(HashMap<String, User> map, UserHashMap uhm) throws
IOException {

                String username;
                sent.askUsername();
                String response = sc.nextLine();
                if (map.containsKey(response)) { // check if this username
                                                                        // already exists
                        sent.askIfOld(); // is new
                        String res = sc.nextLine();

                        if (res.contains("Yes") || res.contains("yes")) { // if the user is old just set
                                                                username
                                username = response;
                        } else { // if the user is new tell him/her to set another username
                                sent.takenUsername();
                                String input = sc.nextLine();
                                while (map.containsKey(input)) { // ask for new username till t's
unique
                                        sent.takenUsername();
                                        input = sc.nextLine();
                                }
                                username = input; // when you find unique username just returnit
                        }

                } else { // if the username is new to the system just return it
                        username = response;
                }
                return username;
        }
}
```

Figure A.1.: The code of the Start state implementation

A.2. Survey questions

The questions and possible answers of the survey questionnaire are presented in the following pages. Conditions to present some of the answers are also outlined.

Movie Recommendation and Information Dialog System Evaluation

Welcome!

If you are looking at this survay, you want to help us evaluate a new module for our social dialog system. The new module creates a user account for you with your movie, actor and genre preferences and then uses this to give you information about your favourite or recommend you new movies.

There are 15 questions in this survey

Demographic Data and General Information

[]Please indicate your gender. *

Please choose **only one** of the following:

- ○ Female
- ○ Male

[]What's your age? *

Please choose **only one** of the following:

- ○ Younger than 18
- ○ 19
- ○ 20
- ○ 21
- ○ 22
- ○ 23
- ○ 24
- ○ 25
- ○ 26
- ○ 27
- ○ 28
- ○ 29
- ○ 30
- ○ 31-35
- ○ 36-40
- ○ 41-45
- ○ 46-50
- ○ older then 50

[]How often do you watch movies? *

Please choose **only one** of the following:

- ○ Every day
- ○ Twice or more times a week
- ○ Every week
- ○ Twice or more times a month
- ○ Every month
- ○ Not every month

[]How often do you use other recommender systems? If you use other recommender systems please write which do you use in the box. *

Please choose **only one** of the following:

- ○ Very frequently
- ○ Frequently
- ○ Infrequently
- ○ Very infrequently
- ○ Do not use

Make a comment on your choice here:

Recommender systems are systems that use your personal data and prefenrences to recommend you items interested for you. I.e. the book recommendation of Amazon, the videos recommendation of YouTube etc.

Satisfaction after the first run of the system

[]How interesting was the information for you? *

Please choose the appropriate response for each item:

	Very boring	Boring	So-so	Interesting	Very interesting
How interesting were the recommended movies for you?	○	○	○	○	○
How interesting was the additional information about your favourite movies/actors?	○	○	○	○	○

[]Was the given information new for you? *

Please choose **only one** of the following:

○ Yes

○ No

○ To some extent

[]How natural does the output of the system seem to you? *

Please choose the appropriate response for each item:

	Very artificial	Artificial	Somewhat artificial and natural	Natural	Very natural
It was...	○	○	○	○	○

[]Why did the output seem artificial?

Only answer this question if the following conditions are met:
Answer was 'Very artificial' *or* 'Artificial' at question '7 [output]' (How natural does the output of the system seem to you? (It was...))

Please write your answer here:

Satisfaction after the second run of the system

[]How interesting was the information for you? *

Please choose the appropriate response for each item:

	Very boring	Boring	So-so	Interesting	Very interesting
How interesting were the recommended movies for you?	○	○	○	○	○
How interesting was the additional information about your favourite movies/actors?	○	○	○	○	○

[]Was the given information new for you? *

Please choose **only one** of the following:

- ○ Yes
- ○ No
- ○ To some extent

[]Did the system use the information from the first run? (to recommend you or give you information) *

Please choose **only one** of the following:

- ○ Yes
- ○ No
- ○ To some extent

[]Did the system recommend you new movies or the same as in the first run? *

Please choose **only one** of the following:

- ○ New
- ○ To some extent new
- ○ The same

[]Did the system give you new information or the same as in the first run? *

Please choose **only one** of the following:

- ○ New
- ○ To some extent new
- ○ The same

Overall satisfaction

[]Do you agree with the following statements: *

Please choose the appropriate response for each item:

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I am satisfied with the system.	○	○	○	○	○
I would continue to use the system.	○	○	○	○	○
I would recommend the system to my friends.	○	○	○	○	○
I can imagine people using the system on daily basis.	○	○	○	○	○

[]Comments

Please write your answer here:

Here you can leave further comments and ideas. Think about:

- Did you like the idea?
- What you like and dislike in the system?
- What did you expect from the system?
- What would you change in the system?

Thank you for your support!

😎

Kalina

Submit your survey.
Thank you for completing this survey.