TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of computer science and cybernetics

Department of Mathematical Informatics

Master thesis

University program Artificial Intelligence

RECOMMENDER DIALOG SYSTEM DEVELOPMENT

2nd year Master's student

Shevchenko Maksym Oleksiiovych ______________

Research advisor:

professor

Marchenko Oleksandr Oleksandrovych ______________

I hereby confirm that this work
does not contain any content
from other sources without
corresponding references.
Student _____________

Kyiv – 2023

SYNOPSIS

74 pages of work, 46 images, 77 reference sources.

CHATBOT, DIALOG SYSTEM, DOMAIN-STATE ARCHITECTURE, CHATGPT, NATURAL LANGUAGE GENERATION, NATURAL LANGUAGE UNDERSTANDING, PYTHON, RECOMMENDATION SYSTEM.

Object of work is a process of generating recommendations based on a user's request using the software application "Dialog system for movie recommendation". **Subject of work** is a movie recommendation software application.

Goal of work is to develop an application for recommendation dialog system.

Methods: natural language processing, software development. **Development tools**: Python programming language, integrated development environment Visual Studio Code, SQLite relational database, GitHub for development hosting and version control.

Results and novelty: a study of modern dialog systems was conducted, and a software application "Dialog system for movie recommendation" was developed. A detailed market research was conducted to ensure the relevance of the developed product.

Software application "Dialog system for movie recommendation" can be used as a standalone application or integrated into modern streaming services.

TABLE OF CONTENT

INTRODUCTION

Dialogue systems, also known as conversational agents, are computer systems that can interact with humans using natural language. They have become increasingly popular and important in natural language processing and real-life applications, such as customer service, education, entertainment, health care, and human-robot interaction [1].

Dialogue systems can enable natural and intuitive communication between humans and computers by using different modalities, such as text, speech, graphics, haptics, gestures, etc. This can improve user satisfaction and efficiency, as well as reduce the cognitive load and errors. For example, dialogue systems can provide a more user-friendly interface for complex tasks, such as booking a flight or ordering food. They can also adapt to different user preferences and contexts, such as language, culture, and emotion.

Also, dialogue systems can advance the research and development of natural language processing and artificial intelligence by using deep learning models, data augmentation methods, and evaluation metrics. These methods can help improve the performance and robustness of dialogue systems in various aspects. For example, deep learning models can learn from large amounts of data and generate natural and coherent responses. Data augmentation methods can enrich the data with synthetic or paraphrased examples to increase the diversity and coverage of the data. Evaluation metrics can measure the quality and effectiveness of dialogue systems based on different criteria and dimensions.

Dialogue systems can be classified into two main types: task-oriented and open-domain.

Task-oriented dialogue systems aim to help users accomplish specific goals or tasks, such as booking a flight, ordering food, or checking the weather.

Open-domain dialogue systems aim to engage users in casual and general conversations, such as chatting about hobbies, interests, or opinions.

Developing dialogue systems is a challenging task that involves many subtasks and components, such as speech recognition, natural language understanding, dialogue management, natural language generation, and speech synthesis. Moreover, dialogue systems need to handle various issues and requirements, such as multimodality, scalability, robustness, personalization, and ethics.

CHAPTER 1. THEORETICAL ASPECTS OF DIALOG SYSTEMS

1.1 Characteristics of Human Dialogue

The act of conversation between two humans is a highly intricate and complex joint activity that requires a certain level of understanding before we can even consider designing a conversational agent to interact with humans. In order to achieve this understanding, it is important to analyze how humans converse with one another. To illustrate this point, let us examine some of the phenomena that can occur in a conversation between a human travel agent and a human client, which is partially depicted in Figure 1.1.1.

C_1: . . . I need to travel in May.
A_2: And, what day in May did you want to travel?
C_3: OK uh I need to be there for a meeting that's from the 12th to the 15th.
A_4: And you're flying into what city?
C_5: Seattle.
A_6: And what time would you like to leave Pittsburgh?
C_7: Uh hmm I don't think there's many options for non-stop.
A_8: Right. There's three non-stops today.
C_9: What are they?
A_{10}: The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.
C_{11}: OK I'll take the 5ish flight on the night before on the 11th.
A_{12}: On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.
C_{13}: OK.
A_{14}: And you said returning on May 15th?
C_{15}: Uh, yeah, at the end of the day.
A_{16}: OK. There's #two non-stops . . . #
C_{17}: #Act. . . actually #, what day of the week is the 15th?
A_{18}: It's a Friday.
C_{19}: Uh hmm. I would consider staying there an extra day til Sunday.
A_{20}: OK. . . OK. On Sunday I have . . .

Figure 1.1.1 – Humans conversation example

The figure presents a small part of a phone conversation between a human travel agent [2], denoted as A, and a human client, denoted as C. It is worth noting that the conversation includes instances of overlaps in speech, which are framed by the symbol

in passages A16 and C17. These phenomena provide us with valuable insights into how humans communicate with one another, which is essential information for developing effective conversational agents.

In a conversation, participants take **turns** to contribute to the dialogue, as if playing a game where each player takes a turn. These turns, such as C1, A2, and C3 in Fig. 1.1.1, consist of a single contribution from one speaker. A conversation can have numerous turns, with the example in Fig. 1.1.1 having a total of 20 turns. The length of a turn can vary, ranging from a single word (C13) to multiple sentences (A10).

The structure of turns is crucial in spoken dialogue, as it affects when a system should start or stop talking. If a client interrupts the system, as in A16 and C17, the system must know to stop talking and be aware that the user may be making a correction. Similarly, in a conversation, speakers typically start their turns immediately after the other speaker finishes, without a long pause. This is because people can usually detect when the other person is about to finish talking. Spoken dialogue systems must also be able to detect when a user has finished speaking so that they can process the utterance and respond. This task, known as endpointing or endpoint detection, can be challenging due to noise and the fact that people may pause in the middle of turns.

One important idea about conversation is that every utterance in a dialogue is a kind of action that the speaker is doing. These actions are often called **speech acts** or dialog acts: here’s one way of dividing them into 4 main categories [3]:

- Constatives: making the speaker responsible for something being true (reporting, asserting, verifying, rejecting, contradicting, informing)
- Directives: trying by the speaker to make the addressee do something (suggesting, questioning, prohibiting, welcoming, commanding, asking)
- Commissives: making the speaker obliged to do something in the future (pledging, intending, swearing, wagering, challenging)

- Acknowledgments: showing the speaker’s attitude towards the hearer regarding some social action (apologizing, greeting, thanking, accepting an acknowledgment)

When a user tells a conversational system or a person to "Play The Godfather," they are performing a DIRECTIVE speech act. Asking a question that seeks information about a movie, such as "Who directed Inception?" is also a type of DIRECTIVE speech act. In contrast, when a user makes a statement that describes their opinion of a movie, such as "I think The Shawshank Redemption is a masterpiece," they are performing a CONSTATIVE speech act. An ACKNOWLEDGMENT speech act is performed when a user expresses gratitude to the system or individual, such as saying "Thanks for the recommendation" after receiving movie suggestions. The speech act serves as an important component of the speaker or writer’s intention in conveying their message.

Grounding is a critical component of successful communication between individuals. In a conversation, grounding refers to the process of establishing mutual understanding between the speaker and the listener, and verifying that both parties share a common understanding of the topic being discussed [4]. Grounding helps to prevent misunderstandings and ensure that communication is effective.

Grounding can occur in a variety of ways. In some cases, individuals may use verbal cues, such as repeating what the other person has said, or explicitly acknowledging their understanding with a phrase like "I see." For example, in a conversation about a project deadline, a team member might say "So, we need to finish this report by next Friday, is that correct?" to ground the other team members.

In some situations, technology can help with grounding as well. For instance, in a virtual meeting, chat messages can be used to clarify understanding, or participants can use "thumbs up" or "thumbs down" emojis to indicate their level of agreement or disagreement. Similarly, in an online gaming environment, players might use in-game chat to confirm that they are working together effectively to achieve a common goal.

Additionally, speakers can use cues to indicate that they have understood the previous utterance, such as using a transitional phrase like "and" to indicate the next question or topic is related to the previous one.

Subdialogues are smaller conversations or discussions that occur within a larger exchange or conversation. They may be initiated by one speaker in response to a comment or question from another speaker, or they may arise spontaneously as a result of the topic being discussed.

Let's see this dialog example:

Person A: I'm thinking of going on a road trip this summer.

Person B: That sounds like fun. Where do you want to go?

Person A: I was thinking of driving down the Pacific Coast Highway.

Person B: Oh, I've driven that before. What parts of it do you want to see?

Person A: I want to see the Big Sur coastline and the Redwood forests.

Person B: Ah, those are both great. When were you thinking of going?

In this example, the subdialogue occurs between Person A and B when Person B asks for more details about Person A's planned road trip. The subdialogue helps to clarify the specific route and destinations that Person A is interested in.

Subdialogues are an important component of dialogue structure because they can add depth and complexity to a conversation. They may be used to explore different perspectives, clarify misunderstandings, or delve into specific details related to the topic at hand.

When structuring a dialogue that includes subdialogues, it is important to consider the flow of the conversation and how the subdialogues fit into the larger context. This can be accomplished by clearly signaling the start and end of subdialogues, using verbal or nonverbal cues to indicate transitions between speakers or topics, and ensuring that each subdialogue has a clear purpose and contributes to the overall conversation.

Communication is not just about the words we use, but also the meaning behind them. In dialogue, we often use **inference and implicature** to convey deeper meanings that are not explicitly stated. Inference and implicature are important components of communication that help us understand what the speaker is really trying to say.

Inference is the process of drawing conclusions based on available evidence. In dialogue, it refers to the process of inferring the meaning of the speaker's words based on the context and the speaker's intentions. For instance, consider the following dialogue:

Person A: "I'm going to the store."

Person B: "Can you pick up some milk for me?"

In this dialogue, Person B infers that Person A is going to the store and, therefore, is in a position to pick up some milk for them. This inference is based on the available evidence in the conversation, namely, that Person A has mentioned going to the store.

Implicature, on the other hand, refers to the meaning conveyed by the speaker through indirect means. Implicature is not explicit in the words spoken but is instead implied through context, tone, or other nonverbal cues. For instance, consider the following dialogue:

Person A: "I'm fine."

Person B: "Okay, but you look upset."

In this dialogue, Person A's response "I'm fine" is an example of implicature. Although the words suggest that everything is okay, Person B infers that Person A is upset based on other cues such as tone of voice or facial expression.

Dialogue systems need to be able to infer meaning from the context and use implicature to respond effectively. This is a significant challenge in natural language processing, as the subtleties of human dialogue can be challenging to interpret accurately.

The intricacies of human conversations such as speech acts, dialogue structure, initiative, grounding, turns, and implicature are some of the key factors that make it challenging to create dialogue systems capable of holding natural conversations with humans. Due to these challenges, there is ongoing research in the field of dialogue systems to address these issues.

1.2 Rule-based chatbots

Chatbots are considered the most basic type of dialogue systems. Their main objective is to imitate casual, unstructured conversations that are typical of human-to-human interactions.

Rule-based chatbots are a type of chatbot that uses a predefined set of rules to determine how to respond to user input [5]. These rules are often based on a decision tree or a series of if-then statements, which determine the bot's response based on the user's input. Rule-based chatbots can be effective for simple tasks and questions, but they have limitations when it comes to more complex conversations or unpredictable user inputs. In this article, we will explore the advantages and disadvantages of rule-based chatbots, how they work, and some examples of their use cases.

ELIZA is one of the earliest and most famous examples of a rule-based chatbot [7]. It was developed in the 1960s by Joseph Weizenbaum, a computer scientist at MIT. The goal of ELIZA was to simulate a conversation with a psychotherapist by using natural language processing techniques to understand and respond to user input.

ELIZA worked by using pattern matching and substitution rules to generate responses (Fig. 1.2.1). It would analyze the user's input and try to identify keywords or phrases that matched its pre-defined patterns. For example, if a user mentioned feeling sad, ELIZA might respond with "Can you tell me more about why you feel sad?" ELIZA's responses were designed to reflect back the user's input, using open-ended questions to encourage the user to continue talking and exploring their feelings.

```
(0 YOU 0 ME) [pattern]
->
(WHAT MAKES YOU THINK I 3 YOU) [transform]
```

Figure 1.2.1 – Example of ELIZA rule

ELIZA has a pattern/rule for each word that it can find in what the user says. Fig. 1.2.2 shows how this works. The key to the algorithm is the specific changes that happen for each word that it recognizes.

```
function ELIZA GENERATOR(user sentence) returns response

Find the word w in sentence that has the highest keyword rank
  if w exists
      Choose the highest ranked rule r for w that matches sentence
      response ← Apply the transform in r to sentence
      if w = 'my'
         future ← Apply a transformation from the 'memory' rule list to sentence
         Push future onto memory queue
   else (no keyword applies)
    either
      response ← Apply the transform for the NONE keyword to sentence
    or
      response ← Pop the oldest response from the memory queue
 return(response)
```

Figure 1.2.2 – Basic diagram of how ELIZA works

ELIZA also has a smart way of remembering the last thing the user said in the conversation above. If "my" is the most important word, ELIZA will pick a transform from the MEMORY list at random, change the sentence with it, and keep it on the line.

```
(MEMORY MY
  (0 MY 0 = LETS DISCUSS FURTHER WHY YOUR 3)
  (0 MY 0 = EARLIER YOU SAID YOUR 3)
  (0 MY 0 = DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR 3
```

Figure 1.2.3 – Example of ELIZA memory queue

On the other hand, ELIZA responds with something vague like "PLEASE CONTINUE", "HOW FASCINATING", or "I UNDERSTAND" if it doesn’t find any keyword.

ELIZA’s simple yet effective approach to natural language processing made it a popular tool for researchers and developers in the early days of AI. It inspired the development of many other chatbots and conversational agents over the years, and its legacy can still be seen in modern chatbot systems.

While ELIZA was groundbreaking in its time, it also had limitations. Its rule-based approach meant that it could only respond to user input that matched its pre-defined patterns, making it vulnerable to errors and misunderstandings when faced with unpredictable or ambiguous inputs. Despite these limitations, ELIZA remains an important milestone in the development of natural language processing and conversational AI.

1.3 Corpus-based chatbots

Corpus-based chatbots are a type of chatbot that use a large collection of natural language data (corpus) to learn how to interact with users [11]. They do not rely on predefined rules or templates, but rather use machine learning techniques to generate responses based on the user’s input and the context of the conversation [6]. Corpus-based chatbots can handle more complex and diverse situations than rule-based chatbots, but they also require more data and computational resources to train and run [12].

The **retrieval** way of answering is to treat what the user says as a query q, and our task is to find and repeat some suitable turn r as the answer from a collection of conversations C. Usually C is the data we use to train the system, and we give each turn in C a score as a possible answer to the context q and pick the one with the highest score. The scoring measure is similarity: we choose the r that is closest to q [2].

Encoder-decoder models produce each token of a response by conditioning on the encoding of both the entire query and the response up to that point [2]. This can be thought of as a machine learning equivalent of ELIZA, where the system learns from a dataset to generate answers based on questions. The generator and retriever methods for response generation are illustrated on Figure 1.3.1. Typically, the generator architecture includes a longer context by incorporating not only the user's current input but also the entire conversation history.

In the response by retrieval method, the chatbot selects a response by identifying the turn in the dataset that has the highest dot product with the user's input. In contrast, the response by generation method employs an encoder-decoder architecture to generate the response.

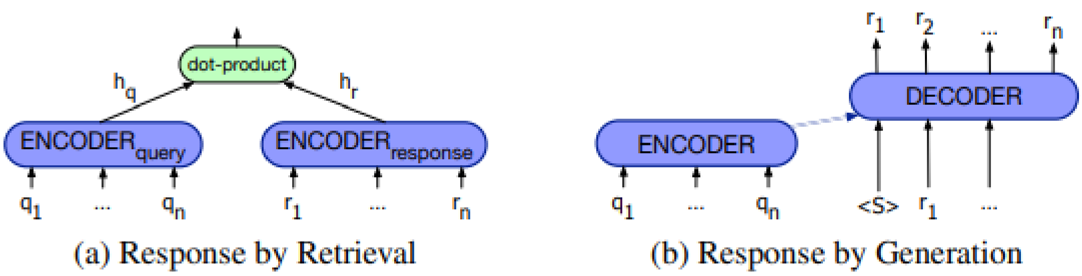


Figure 1.3.1 – Two ways of creating answers for a neural chatbot

1.4 Frame-based dialogue systems

Frame-based dialogue systems are a type of conversational agent that use structured knowledge representations known as frames to understand and generate natural language conversations. These systems have become increasingly popular in recent years due to their ability to handle complex and dynamic dialogue scenarios.

At their core, frame-based dialogue systems are built around a set of predefined frames [1], which are essentially semantic structures that define the concepts, relationships, and actions relevant to a specific domain. For example, a frame for a restaurant might include information about menu items, prices, seating arrangements, and reservation policies. Each frame contains a set of slots, which represent specific pieces of information associated with the frame.

For instance, in the context of a travel domain, the slots might be of types such as city, date, airline, or time. This means that the system is only capable of accepting values that correspond to those particular types, such as "Kyiv" for a city slot, or "March 1st, 2023" for a date slot. In the restaurant example, slots might include the name of the restaurant, the location, and the cuisine type. By using semantic types to constrain the possible values for each slot, the system can ensure that it accurately interprets the user’s input and provides relevant responses.

Slot	**Type**	Question Template
ORIGIN CITY	city	“From what city are you leaving?”
DESTINATION CITY	city	“Where are you going?”
DEPARTURE TIME	time	“When would you like to leave?”
DEPARTURE DATE	date	“What day would you like to leave?”
ARRIVAL TIME	time	“When do you want to arrive?”
ARRIVAL DATE	date	“What day would you like to arrive?”

Figure 1.4.1 – Frame examples in travel domain system

When a user initiates a conversation with a frame-based dialogue system, the system uses natural language processing techniques to identify the user’s intent and map it to the appropriate frame. For example, if the user says "I’m looking for a sushi restaurant in the downtown area," the system would use entity recognition to identify the user’s location and cuisine preferences and then match those entities to the corresponding slots in the restaurant frame.

Once the system has identified the relevant frame and filled in the necessary slots, it can generate a response that provides the user with the information they requested. For example, the system might say "There are several sushi restaurants in the downtown area. Would you like me to provide a list of options?"

One of the key advantages of frame-based dialogue systems is their ability to handle complex and dynamic dialogue scenarios. Because the system is built around a set of well-defined frames and slots, it can handle a wide range of user requests and follow-up questions without becoming confused or overwhelmed. For example, if the user asks "What are the prices like at the sushi restaurants in the downtown area?", the system can easily retrieve the relevant information from the restaurant frame and provide an appropriate response.

Another advantage of frame-based dialogue systems is their ability to learn from user interactions over time. By analyzing user behavior and feedback, the system can continually refine its understanding of the domain and improve its ability to handle user requests. This can lead to a more personalized and effective user experience over time.

GUS (**Genial Understander System**) is a dialog system that has been developed by the University of Rochester for natural language processing tasks [16]. It was designed to be a general-purpose system that can be used for a wide range of applications, including chatbots, virtual assistants, and customer service agents.

One of the key features of the GUS dialog system is its use of natural language understanding (NLU) techniques to parse and interpret user input. The system is able to analyze the structure of a sentence and identify the semantic relationships between the various elements. This allows GUS to generate a precise and accurate understanding of what the user is trying to say, even if the input is complex or ambiguous.

To achieve this level of accuracy, GUS relies on a number of sophisticated NLU components, including syntactic parsing, semantic role labeling, and named entity recognition. These techniques allow the system to identify the relationships between words in a sentence and assign them to specific semantic roles, such as subject, object, or location.

Once the system has generated a precise understanding of the user's intent, it can then generate an appropriate response. GUS uses a set of predefined frames that define the domain-specific concepts and relationships that are relevant to a particular task. These frames include a set of slots, each of which represents a specific piece of information that the system needs to know in order to generate an appropriate response.

For example, if a user asks GUS to find a nearby Italian restaurant, the system would use its NLU components to identify the user's location and cuisine preferences and map them to the corresponding slots in the restaurant frame. It could then generate a response that provides the user with a list of nearby Italian restaurants.

The system fills all the slots that the user's answer matches, and then keeps asking questions to fill the rest of the slots, skipping questions that are already filled. The GUS architecture also has rules that link slots. For example, a rule that links the DESTINATION slot for the plane booking frame, after the user has given the destination, might automatically put that city as the default StayLocation for the hotel booking frame that is related. Or if the user gives the DESTINATION DAY for a short trip the system could automatically put the ARRIVAL DAY.

One of the key advantages of the GUS dialog system is its flexibility and scalability. Because it is designed to be a general-purpose system, it can be easily customized and adapted to a wide range of applications and domains. This makes it an ideal tool for developers and businesses that need to create conversational agents that can handle complex and dynamic dialogue scenarios.

Overall, the GUS dialog system represents a significant advancement in the field of natural language processing and conversational AI. Its sophisticated NLU techniques and flexible frame-based architecture make it a powerful tool for creating conversational agents that can understand and respond to a wide range of user requests in a natural and intuitive way. As the field of conversational AI continues to evolve, it is likely that we will see even more sophisticated and effective systems like GUS emerge in the years to come.

1.5 Dialog-state architecture

A common type of task-oriented dialogue system nowadays uses an advanced form of the frame-based architecture known as the dialogue-state or belief-state architecture. It is based on the concept of maintaining a persistent "dialog state" throughout a conversation, which represents the current state of the conversation and the information that has been exchanged. The main parts of a typical dialogue-state system are shown in Figure 1.5.1 [43].

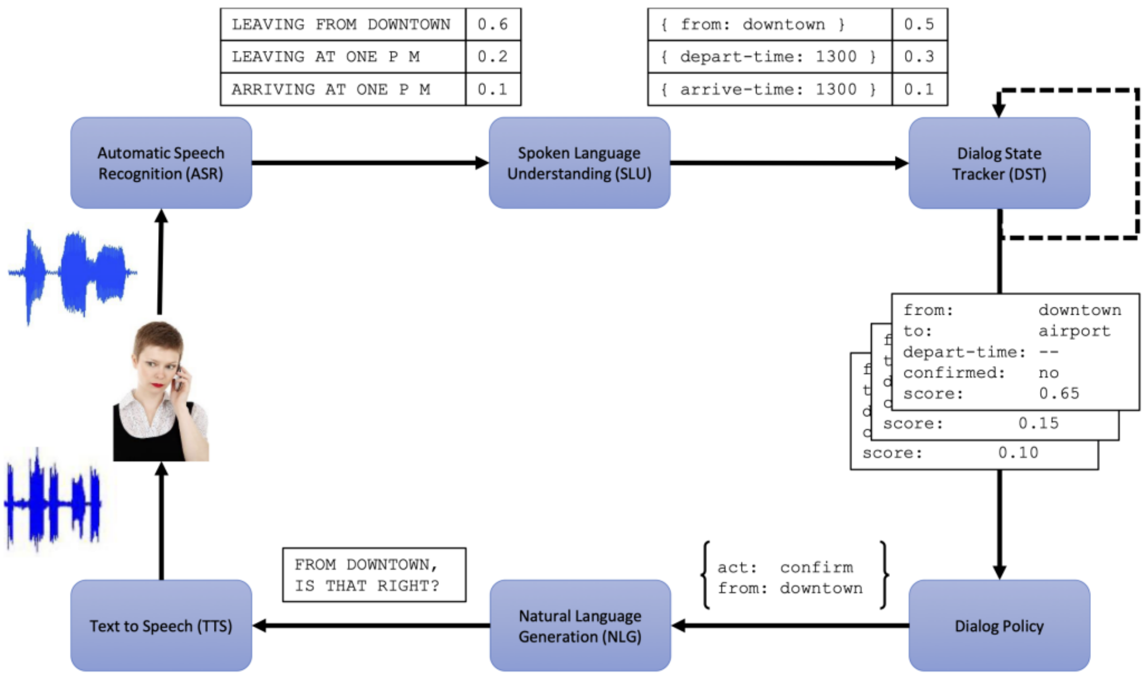


Figure 1.5.1 – Task-oriented dialog-state architecture with conversation example

A **dialog act** is a way of representing the communicative function of an utterance in a conversation, such as greeting, requesting, informing, etc. A dialog act can also include grounding acts, which are signals that indicate the level of understanding or agreement between the speakers. In dialog-state architecture, dialog acts are used to track the user's intentions and goals, as well as the system's responses and actions. Dialog acts can be derived from user speech, natural language understanding output, or context from previous turns. Dialog acts can also be used to generate system utterances based on a dialog policy and a reward function [46]. Figure 1.5.2 [44] shows how this works.

	Utterance	Dialogue act
U:	Hi, I am looking for somewhere to eat.	hello(task = find,type=restaurant)
S:	You are looking for a restaurant. What type of food do you like?	confreq(type = restaurant, food)
U:	I'd like an Italian somewhere near the museum.	inform(food = Italian, near=museum)
S:	Roma is a nice Italian restaurant near the museum.	inform(name = "Roma", type = restaurant, food = Italian, near = museum)
U:	Is it reasonably priced?	confirm(pricerange = moderate)
S:	Yes, Roma is in the moderate price range.	affirm(name = "Roma", pricerange = moderate)
U:	What is the phone number?	request(phone)
S:	The number of Roma is 385456.	inform(name = "Roma", phone = "385456")
U:	Ok, thank you goodbye.	bye()

Figure 1.5.2 – Dialog acts example in a sample dialog

Slot filling is a key task in dialog state tracking that aims to extract the values of certain predefined slots that are relevant to the user's goal and the system's task. For example, in a flight booking system, the slots may include origin, destination, date, time, etc. Slot filling can be done by matching the user utterances and system responses with the ontology, which is a collection of possible slot values for each domain and task [48]. Slot filling can also be done by using neural models that can learn from the dialog context and attention mechanisms [49]. Slot filling is important for dialog state tracking because it helps the system to keep track of the user's preferences and constraints, and to generate appropriate responses and actions [48; 49].

Dialogue state tracking is the task of estimating the user's intentions and goals at each turn of a dialogue, given the previous dialogue history [50]. Dialogue state tracking is essential for dialog-state architecture because it enables the system to manage and generate dialogues that are coherent, relevant, and helpful for the user. Dialogue state tracking usually involves filling in slots that represent the key information and constraints for the task, such as flight booking, restaurant reservation, etc. Dialogue state tracking can also include recognizing the user's dialogue act, such as greeting, requesting, informing, etc. Dialogue state tracking can be done by using rule-based methods, statistical methods, or neural methods [43; 51]. Figure 1.5.3 shows how dialog state tracker connected with other parts of the dialog-state system.

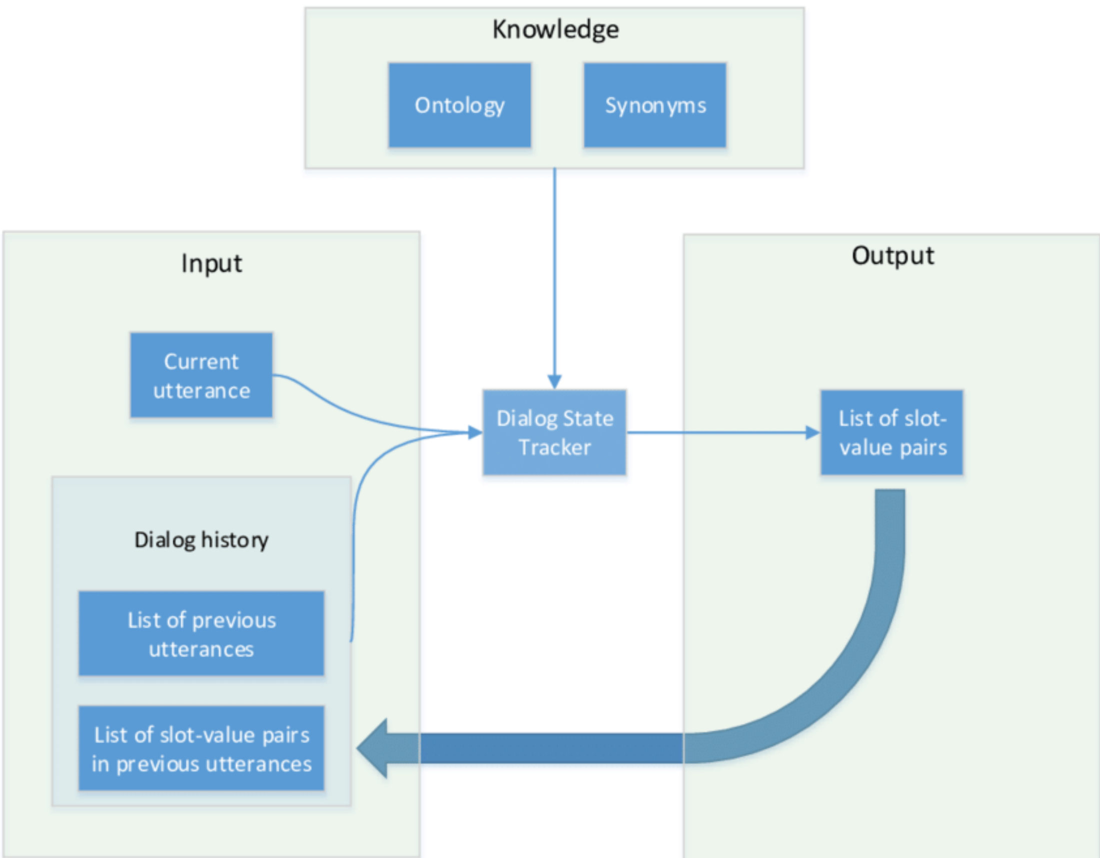


Figure 1.5.3 – Overview of the system for dialog state tracking

Dialogue policy is the component of a dialog-state architecture that decides what action the system should take at each turn of a dialogue, based on the current dialogue state and the system's goal. Dialogue policy can be rule-based, where the system follows a predefined set of rules or scripts to generate actions, or data-driven, where the system learns from data using reinforcement learning, supervised learning, or imitation learning. Dialogue policy can also be modular, where the system has separate policies for different domains or tasks, or end-to-end, where the system has a single policy for the whole dialogue. Dialogue policy is important for dialog-state architecture because it determines how the system interacts with the user and guides the dialogue towards a successful outcome [52].

Natural language generation is the process of producing natural language utterances from a semantic representation, such as a dialogue act, a dialogue state, or a system action. Natural language generation is the final component of a dialog-state architecture that interacts with the user and conveys the system's response [54]. Natural language generation can be template-based, where the system uses predefined templates to fill in the slots with values, or data-driven, where the system learns from data using neural models, such as recurrent neural networks, convolutional neural networks, or transformers. Natural language generation can also be modular, where the system has separate generators for different domains or tasks, or end-to-end, where the system has a single generator for the whole dialogue. Natural language generation is important for dialog-state architecture because it determines how the system expresses its actions and information in a natural, fluent, and informative way [55].

CHAPTER 2. MODERN DIALOG SYSTEMS

Modern dialog systems have revolutionized the way we interact with computers and digital devices. These systems use natural language processing (NLP) and machine learning (ML) techniques to understand human language and provide useful responses.

There are many dialog systems that have been developed for different purposes and domains. Some of the most popular dialog systems are:

- Digital voice assistants, such as Apple Siri, Amazon Alexa, Google Assistant, and Yandex Alice. These are goal-oriented dialog systems that can help users perform various tasks on their devices or on the web, such as setting reminders, making calls, searching for information, or controlling smart home devices [18].
- ChatGPT, an open-domain dialog system that can engage users in casual and general conversations on any topic, using a large-scale neural network model trained on massive amounts of conversational data [19].
- Bing AI, a hybrid dialog system that can handle both goal-oriented and open-domain dialogs, using a combination of rule-based and data-driven approaches.
- ParlAI, a framework for developing and evaluating dialog systems that contains over 80 popular dialog datasets and models [19].
- DSTC, a series of challenges and datasets for dialog state tracking, which is a core component of goal-oriented dialog systems [19].
- Persona-Chat and ConvAI2, datasets and challenges for building open-domain dialog systems that can use personas to personalize and empathize with users [19].

The rise of dialog systems can be attributed to advances in artificial intelligence (AI) and the increasing demand for personalized, on-demand services. Today, users can interact with these systems through voice commands, messaging platforms, and even smart home devices, making it easier than ever to get information and accomplish tasks.

Digital assistants have become ubiquitous in our daily lives, from setting reminders and scheduling appointments to ordering groceries and controlling smart home devices. Google Assistant, for example, has over 1 billion active users worldwide, and Siri is installed on over 1.5 billion Apple devices. These systems use speech recognition technology to understand spoken commands, and ML algorithms to provide personalized recommendations based on a user's preferences and past interactions.

Chatbots, on the other hand, are designed to simulate human conversations and provide instant customer support. ChatGPT and Bing AI are examples of chatbots that use NLP techniques to understand and respond to user queries. These systems can be trained on large datasets of real-world conversations, allowing them to generate natural-sounding responses and learn from past interactions.

In this chapter, we'll explore the various types of modern dialog systems.

2.1 Digital assistants (Google Assistant and Apple Siri)

Google Assistant is a digital assistant developed by Google that uses natural language processing and machine learning algorithms to provide personalized assistance to users. It is available on various devices, such as smartphones, smart speakers, smart displays, laptops, and TVs, and can be activated by voice commands or typing [20].

Google Assistant can perform a variety of tasks, from setting reminders and alarms to controlling smart home devices, making phone calls, and sending text messages. It can also provide information on a wide range of topics, including weather, news, and sports scores, and can even make restaurant reservations and book flights [21].

One of the key features of Google Assistant is its ability to understand natural language and provide personalized responses based on a user's preferences and past interactions. For example, it can learn a user's favorite restaurants and make recommendations based on their previous dining habits.

Google Assistant also integrates with a wide range of third-party apps and services, allowing users to perform tasks and access information across multiple platforms. For example, users can order food from popular delivery services like DoorDash and Grubhub, or check their bank account balance with financial apps like Mint.

Figure 2.1.1 – Conversation with Google Assistant

Google Assistant is constantly evolving, with new features and capabilities added regularly. Some recent updates include the ability to make group audio and video calls, improved voice recognition technology, and the ability to schedule and manage smart home routines.

Siri is a virtual assistant developed by Apple Inc. that uses natural language processing and machine learning to interact with users and perform a wide range of tasks. It is available on various devices, such as iPhones, iPads, Macs, Apple Watches, Apple TVs, and HomePods [22]. It was first introduced in 2011 as a feature of the iPhone 4S, and has since been integrated into other Apple products like the iPad, Apple Watch, and Mac computers.

Siri can be activated by voice commands or by pressing a button on supported devices, and can perform a wide range of tasks, from making phone calls and sending text messages to setting reminders, playing music, and providing directions [23].

One of the key features of Siri is its ability to understand natural language and adapt to a user's individual preferences and usage patterns over time. For example, if a user frequently calls a specific contact or listens to a particular type of music, Siri can learn these preferences and provide personalized recommendations and shortcuts.

Siri also integrates with a wide range of third-party apps and services, allowing users to perform tasks and access information across multiple platforms. For example, users can order food from popular delivery services like Grubhub and Uber Eats, or make reservations through travel apps like Airbnb and Kayak.

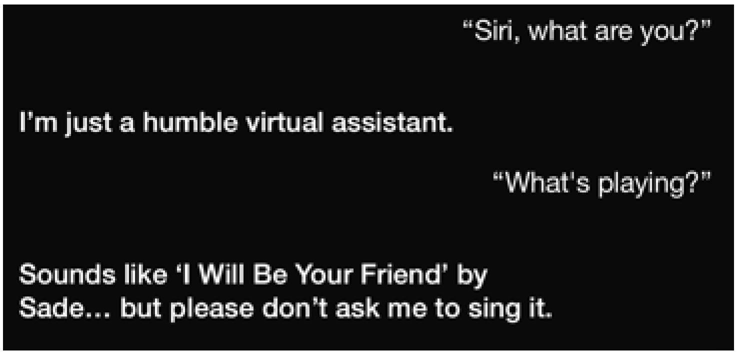


Figure 2.1.2 – Conversation example with Siri

Apple has continued to develop and improve Siri over the years, with recent updates adding new features like support for multi-language translation, improved voice recognition technology, and the ability to create custom shortcuts and automation workflows.

Siri and Google Assistant are two of the most popular virtual assistants available today. While they share many similarities, there are also some key differences between the two that set them apart.

1. Natural Language Processing:

Both Siri and Google Assistant use natural language processing to understand and respond to user queries. However, Google Assistant is generally considered to be more advanced in this area, with a greater ability to understand complex sentences and context. Google Assistant also uses Google's powerful search engine to provide more accurate and comprehensive responses to user queries.

2. Platform compatibility:

Google Assistant is available on more devices than Siri, as it can work on both Android and iOS devices, as well as various smart speakers, TVs, and other gadgets. Siri is mainly limited to Apple devices, such as iPhones, iPads, Macs, Apple Watches, Apple TVs, and HomePods.

3. Language support:

Google Assistant supports more languages than Siri, as it can understand and speak up to 30 languages [25], while Siri can only handle 17 languages [26]. However, Siri has more options for voice gender and accent than Google Assistant.

4. Integration with Third-Party Apps:

Both Siri and Google Assistant integrate with a wide range of third-party apps and services, allowing users to perform tasks and access information across multiple platforms. However, Google Assistant has an advantage in this area, with a more open platform that allows for easier integration with third-party developers.

5. Personalization:

Siri is known for its ability to learn a user's preferences and usage patterns over time, providing personalized recommendations and shortcuts. Google

Assistant also offers some degree of personalization, but it is generally not as advanced as Siri's.

6. General knowledge:

Google Assistant is better at answering trivia and factual questions than Siri, as it has a closer integration with Google Search and can provide more detailed and relevant information. Siri often fails to answer such questions or provides incomplete or inaccurate answers.

7. Additional Features:

Both Siri and Google Assistant offer a wide range of features, from setting reminders and alarms to controlling smart home devices and playing music. However, Google Assistant has a broader range of capabilities, including the ability to make phone calls and send text messages on behalf of the user.

In summary, while Siri and Google Assistant share many similarities, there are some key differences between the two. Google Assistant is generally considered to be more advanced in terms of natural language processing and integration with third-party apps, while Siri is known for its ability to provide personalized recommendations and shortcuts. Ultimately, the choice between the two will depend on the user's specific needs and preferences.

2.2 ChatGPT

ChatGPT (Generative Pre-trained Transformer) is a cutting-edge conversational AI system developed by OpenAI. It is built on top of OpenAI's GPT-3.5 and GPT-4 families of large language models (LLMs) and has been fine-tuned using both supervised and reinforcement learning techniques [27].

For better understanding of the ChatGPT complexity and evolution, let's compare different versions of GPT [31]:

1. GPT-1: The first version of GPT was released in 2018 and was based on the Transformer architecture. It had 12 layers, 117 million parameters, and was trained on 40 GB of text from the WebText dataset.

2. GPT-2: The second version of GPT was released in 2019 and was a scaled-up version of GPT-1. It had 48 layers, 1.5 billion parameters, and was trained on 40 GB of text from the WebText dataset. It was considered one of the most advanced language models at the time and generated high-quality and coherent texts on various topics. However, it also raised concerns about its potential misuse and ethical implications, such as generating fake news, spam, or propaganda [32].

3. GPT-3: The third version of GPT was released in 2020 and was a massive leap from GPT-2. It had 96 layers, 175 billion parameters, and was trained on 45 TB of text from the Common Crawl dataset. It was the largest language model ever created and achieved state-of-the-art results on many natural language processing benchmarks and tasks. It also introduced a few-shot learning technique, which allowed it to perform a specific task with just a few examples as input. However, it also had some limitations and challenges, such as generating factual errors, biases, or harmful outputs.

GPT-3.5 model is an improved version of the GPT-3 model. It was created in January 2022 and has three different sizes with 1.3B, 6B and 175B parameters each. Its main feature was to reduce the toxic output to some degree. The following models are in the GPT-3.5 series [28]:

- code-davinci-002 is a base model, so good for pure code-completion tasks
- text-davinci-002 is an InstructGPT model based on code-davinci-002
- text-davinci-003 is an improvement on text-davinci-002

4. GPT-4: The fourth version of GPT was released in 2023 and was a further improvement of GPT-3. It had 192 layers, 350 billion parameters, and was trained on 90 TB of text from the Common Crawl dataset [29]. It also

incorporated more sources of knowledge and data, such as images, videos, audio, tables, graphs, and equations. It was able to generate multimodal texts that were more accurate, diverse, and creative than previous versions. It also addressed some of the safety and ethical issues of GPT-3 by using filters, feedback mechanisms, and transparency tools.

ChatGPT is designed to engage in natural language conversations with users, and has been trained on massive amounts of text data to improve its ability to understand and generate human-like responses. It is capable of answering a wide range of questions on a variety of topics, from trivial and general knowledge to more complex queries that require deeper understanding.

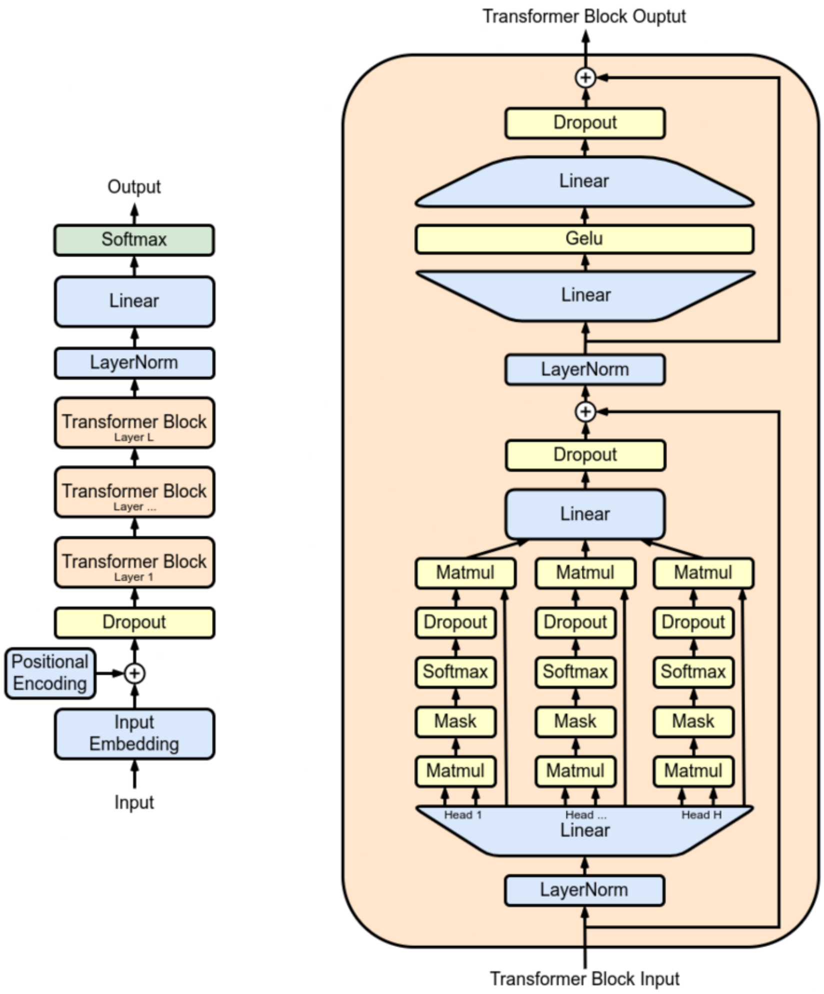


Figure 2.2.1 – Original GPT architecture

One of the key features of ChatGPT is its ability to adapt to a user's conversational style and respond in a way that feels natural and engaging. It can also understand context and follow the flow of a conversation, making it ideal for use in chatbots, virtual assistants, and other conversational interfaces.

OpenAI has made ChatGPT available to developers through its API, allowing them to integrate the system into their own applications and services. This has led to the development of a wide range of innovative chatbot applications that use ChatGPT to provide personalized, human-like interactions with users.

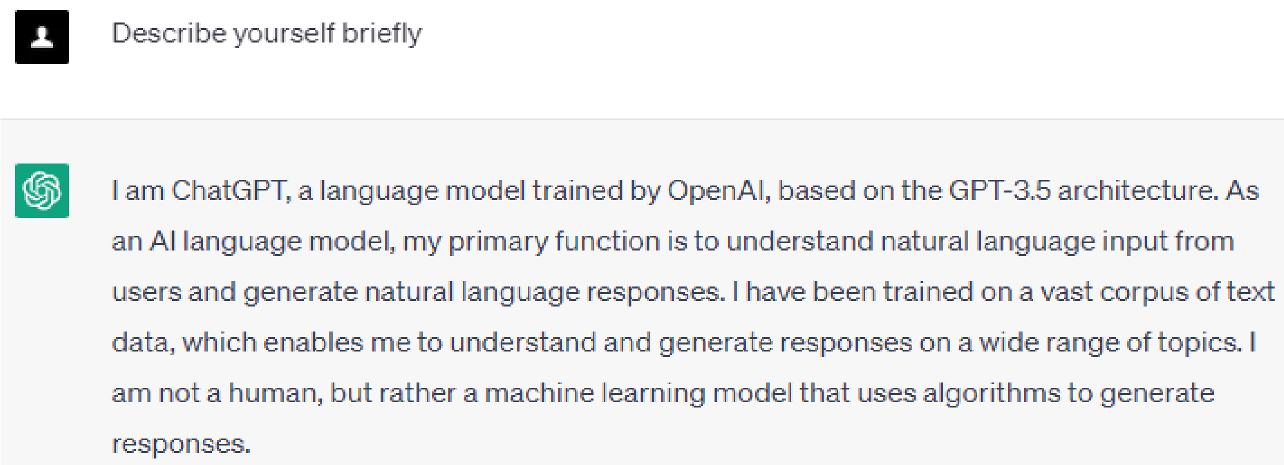


Figure 2.2.2 – Conversation example with ChatGPT (using GPT-3.5)

Overall, ChatGPT is a powerful conversational AI system that represents a significant step forward in the field of natural language processing. Its ability to generate human-like responses and engage in natural language conversations has the potential to transform the way we interact with machines and automate a wide range of tasks and services.

2.3 Bing AI

Bing AI is a suite of artificial intelligence services and tools developed by Microsoft to enhance its Bing search engine and other products. It includes a range of natural language processing capabilities, such as sentiment analysis, named entity recognition, and language translation, as well as machine learning tools for image and speech recognition [37].

One of the key features of Bing AI is its ability to integrate with other Microsoft products, such as Office 365 and Azure, allowing developers to build more advanced applications and services that leverage these AI capabilities. For example, developers can use Bing AI to build chatbots that can understand and respond to user queries in natural language, or to improve the accuracy of speech recognition systems for applications like virtual assistants.

Bing AI also includes features designed to improve the relevance and accuracy of search results. For example, it uses machine learning algorithms to understand the intent behind user queries and surface more relevant content. It can also analyze the content of web pages and other documents to extract key insights and provide more informative search results.

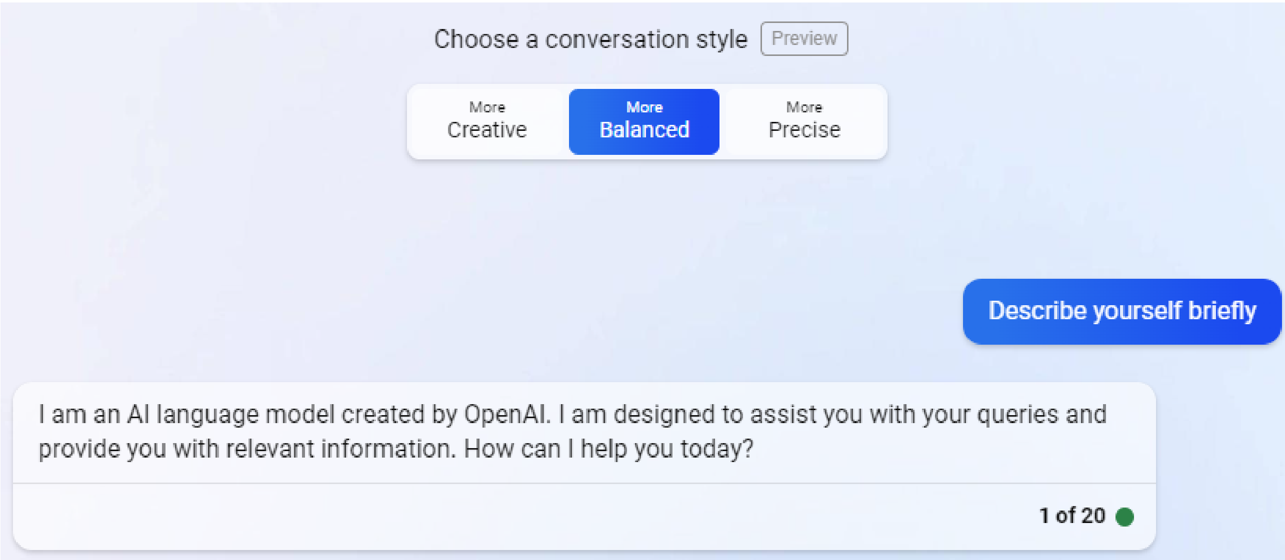


Figure 2.3.1 – Example of the conversation with Bing AI

Overall, Bing AI represents Microsoft's ongoing commitment to advancing the field of artificial intelligence and integrating these capabilities into its products and services [35]. As AI continues to play an increasingly important role in the technology landscape, Bing AI is well-positioned to help developers and organizations leverage these capabilities to create more intelligent and intuitive applications and services.

ChatGPT and Bing AI are both AI-powered chatbots that can interact with users using natural language. However, they have some differences in terms of their features, capabilities, and compatibility. Here are some of the main points of comparison between ChatGPT and Bing AI:

- Purpose: ChatGPT is an open-domain chatbot that can engage users in casual and general conversations on any topic. Bing AI is a hybrid chatbot that can handle both goal-oriented and open-domain dialogs, as well as provide search results and information from the web [39].
- Conversational language: ChatGPT and Bing AI both use the GPT language model developed by OpenAI, but they are running different versions. ChatGPT uses GPT-3.5-turbo, while Bing AI uses GPT-4-Prometheus. ChatGPT is more creative and versatile, but also unpredictable and inconsistent. Bing AI is more accurate and reliable, but also more constrained and formal [40].
- Data accuracy: ChatGPT does not have the ability to index the web in real-time for information, so it is limited to the training data in its model, which dates back to September 2021. ChatGPT also has explicit disclaimers about data inaccuracy and does not provide sources for its responses. Bing AI can access the web and provide more up-to-date and relevant information, as well as cite its sources [41].
- Training Data: ChatGPT was trained on a large corpus of text data, including books, websites, and other sources, whereas Bing AI is trained on a more diverse set of data, including speech and images. This difference in training data can impact the types of tasks each platform is best suited for.
- Mathematical precision: ChatGPT and Bing AI both understand math and can perform calculations, but they have different levels of precision.

ChatGPT can handle complex equations and expressions, but sometimes makes errors or gives approximate answers. Bing AI can handle simple calculations and conversions, but gives exact answers [41].

- Security measures: ChatGPT and Bing AI both have some security measures to prevent misuse and abuse of their systems. ChatGPT requires users to sign up with an account and agree to a code of conduct. It also filters out profanity, personal information, and harmful content. Bing AI requires users to join a waitlist and agree to a term of service. It also uses filters, feedback mechanisms, and transparency tools to ensure safety and ethics [42].
- Accessibility: ChatGPT and Bing AI are both available for anyone who signs up with an account, but they have different levels of accessibility. ChatGPT is only available on the web at chat.openai.com, while Bing AI is available on various platforms and applications, such as Microsoft Edge, Outlook, Word, PowerPoint, Teams, Skype, Bing app, and more.
- Availability: ChatGPT is an open-source platform that can be accessed by developers and researchers, whereas Bing AI is a proprietary product that requires a subscription to access.

CHAPTER 3. DIALOG SYSTEM DEVELOPMENT

3.1 Subject and relevance of the product

Although the first dialog systems appeared quite a while ago, they have become widespread relatively recently. As mentioned in Chapter 2, today there are about 1 billion active users of Google Assistant and about 1.5 billion active users of Siri. This demonstrates the popularity of dialog systems at the present time. However, more interesting statistics can be found in relation to ChatGPT.

According to various sources, ChatGPT has reached 100 million users in January 2023, just two months after launching [56]. It had about 590 million visits in January from 100 million unique visitors [57]. It also set a record for having the fastest-growing user base in history for a consumer application, gaining 1 million users in just 5 days [56]. Around 300 billion words were fed into the system of ChatGPT [58]. The AI receives approximately 10 million daily queries [58].

ChatGPT has become a phenomenon in the social media and technology world, with more than 40 percent of adults in the United States being aware of the program [59]. It has been praised for its ability to generate coherent and diverse responses, as well as its humor and personality.

The above statistics show that users are ready to use dialog systems and use them to solve their needs.

Watching movies can be considered a popular hobby or interest for many people [59]. Watching movies is a form of entertainment that can provide you with enjoyment, relaxation, inspiration, and education. Watching movies is a hobby that can suit anyone who loves stories, visuals, sounds, and emotions. It is a hobby that can enrich your life in many ways and help you discover new aspects of yourself and the world.

According to a social survey [60], 36% and 26% of respondents watched movies and TV series very often, and 18% and 11% often.

YouTube is the world’s largest video-sharing platform that allows users to upload, watch, comment, and share videos on various topics. According to Statista [61; 62], YouTube had 2.3 billion MAU worldwide in 2022 and is projected to have 2.5 billion MAU in 2026. The countries with the most YouTube users are India (467 million), USA (246 million), Brazil (142 million), Indonesia (139 million), and Mexico (81.8 million) [64]. The YouTube mobile app had 866 million MAU worldwide on iPhone and iPad combined in the fourth quarter of 2021 [62].

Netflix is the world’s leading streaming entertainment service that offers a wide variety of movies, shows, documentaries, and original content. According to Statista [63], Netflix had 209 million paid subscribers worldwide as of the second quarter of 2021. The Netflix mobile app had 115 million MAU worldwide on iPhone and iPad combined as of June 2021 [63]. The countries with the most Netflix subscribers are USA (73.9 million), Brazil (20.5 million), UK (14.7 million), Mexico (9.5 million), and India (8.4 million) [64].

Amazon Prime Video is a streaming service that offers movies, shows, documentaries, and original content to Amazon Prime members. According to Statista [65], Amazon Prime Video had 175 million subscribers worldwide as of the first quarter of 2021. The countries with the most Amazon Prime Video subscribers are USA (75 million), Germany (26 million), Japan (25 million), UK (15 million), and India (11 million) [64].

Demand creates supply, so there are a lot of new movies and TV series coming out every day. Netflix has a global library of over 6000 movies and TV shows, with more than 1500 original titles [66]. Amazon Prime Video has a global library of over 8500 movies and TV shows, with more than 200 original titles [66].

Due to the large size of the library, viewers may have difficulties in choosing movies to watch. Figure 3.1.1 shows an example of one of the videos on the YouTube platform with a collections of movies on different platforms recommended for viewing. As you can see, the video has collected almost 4 million views. On YouTube, there are many other videos on a similar topic, which collect a lot of views. This shows that people are looking for recommendations on what to watch next.

Figure 3.1.1 – An example of a video with a collection of films recommended for viewing

The following conclusions can be drawn from the above:

1. People like to watch movies and TV shows
2. People need recommendations because of the abundance of content
3. People actively use dialog systems

The development of a dialog system that could provide movie recommendations could satisfy the above functions and have a clear and familiar user interface.

Thus, the software application "Dialog system for movie recommendation" is relevant and can become popular among users.

3.2 Overview of modern solutions

To understand the need for recommender systems, let’s first consider the functionality of classic platforms. Most modern platforms for watching movies have similar capabilities and differ in content, design and functionality. To systematize the features, we will divide the platforms into two general classes: online cinemas and streaming services.

Online cinema is a service that allows you to watch movies, TV series, cartoons, and TV shows over the Internet. As an example, consider one of the popular Ukrainian online cinemas – **MEGOGO** [67].

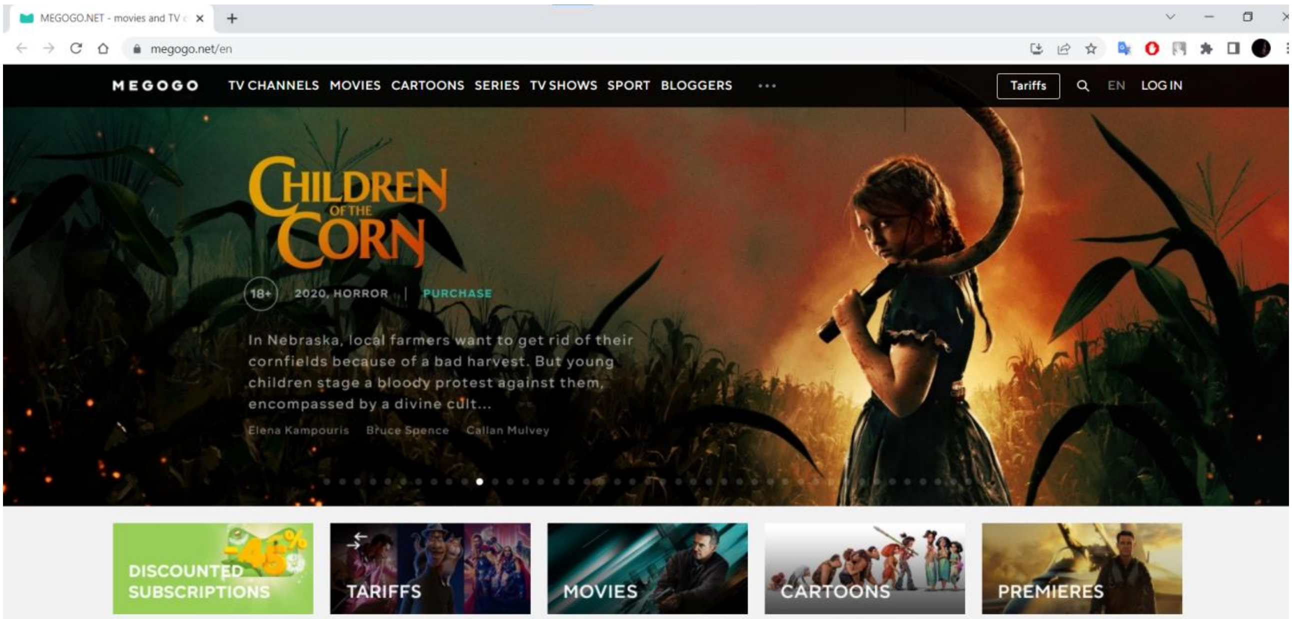


Figure 3.2.1 – The home page of the megogo.net website

MEGOGO.net is a website that offers online cinema services. You can watch movies, TV shows, live sports, audiobooks, and podcasts on megogo.net. The website has content in different languages and genres, such as comedy, drama, action, romance, and more. You can watch some content for free, or you can subscribe to get access to more features and premium content. MEGOGO.net also has an app for Android and iOS devices, as well as for Smart TVs, media players, and gaming consoles.

In addition, the website provides access to online TV channels (Figure 3.2.2).

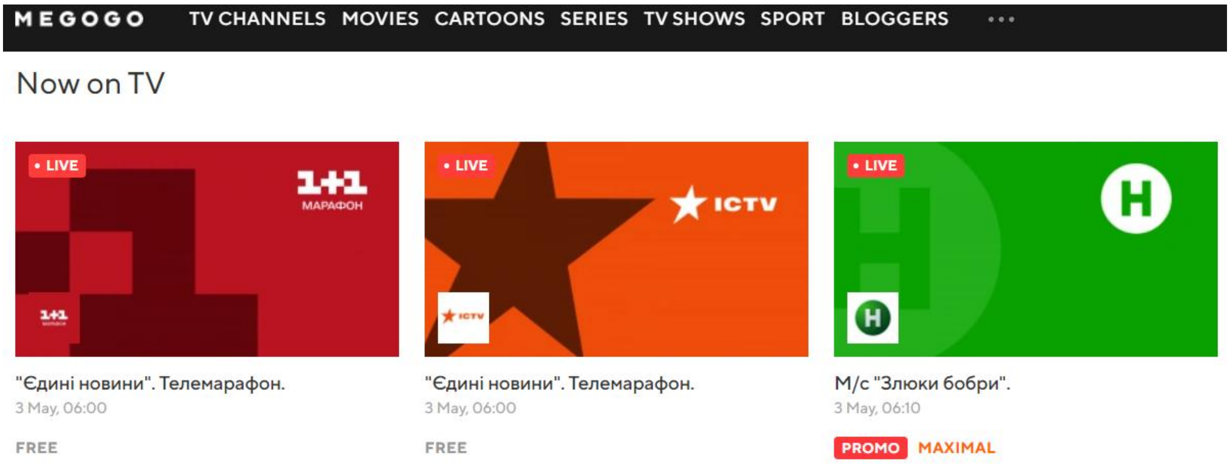


Figure 3.2.2 – A list of some of the TV channels available for viewing on the MEGOGO online cinema

In terms of content search, MEGOGO offers both its own collections (Figure 3.2.3) and the ability to search using certain filters (Figure 3.2.4), such as genre, year of release, and language.

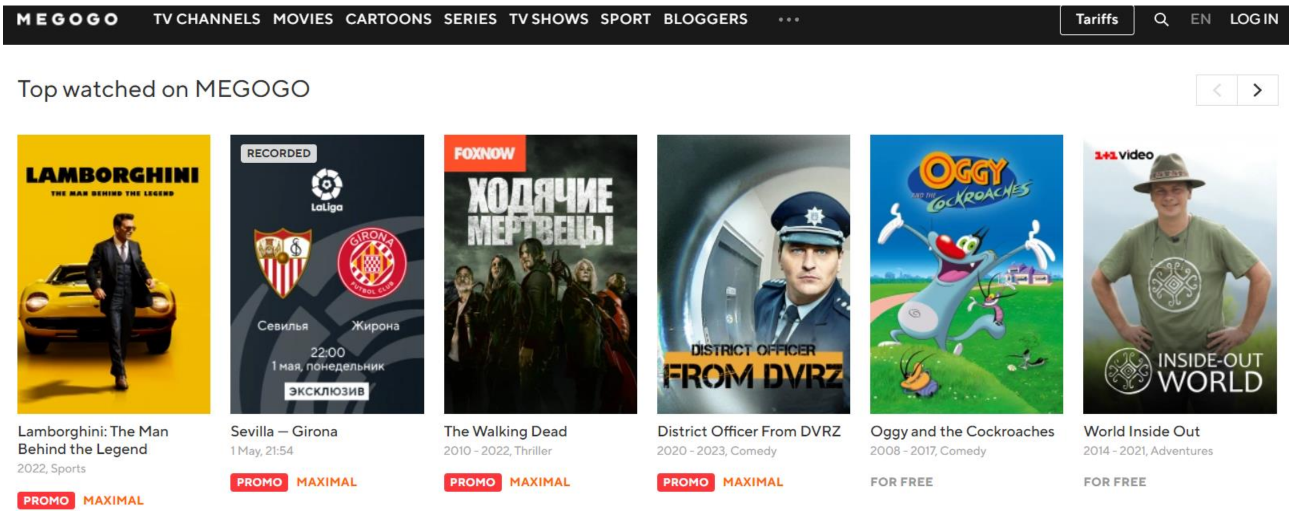


Figure 3.2.3 – An example of the most popular content in the MEGOGO

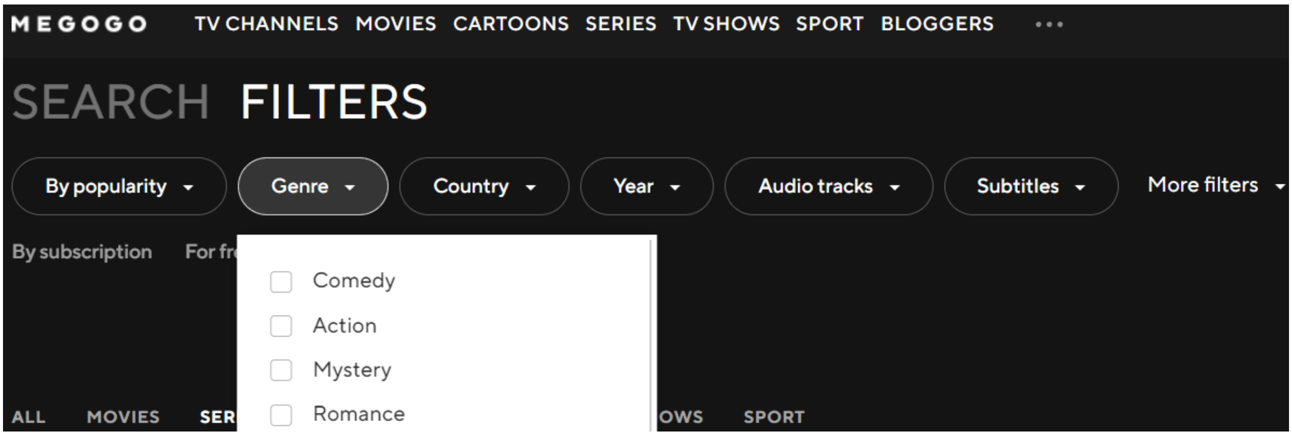


Figure 3.2.4 – An example of search filters in the MEGOGO

Netflix is one of the most popular streaming services. Unlike MEGOGO, it does not provide the ability to watch TV channels, but it has an impressive library of its own unique content.

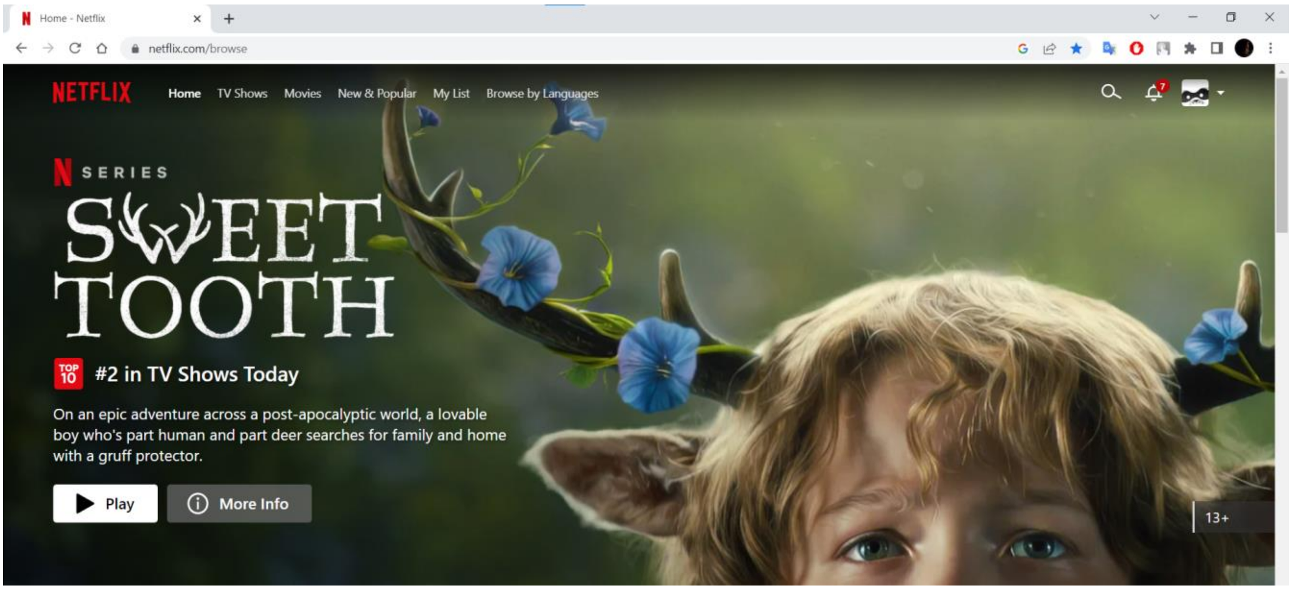


Figure 3.2.5 – Home page of netflix.com

Other similar functional features include the availability of collections of popular content (Fig. 3.2.6) and the ability to search with basic filters such as genre and actors (Fig. 3.2.7).

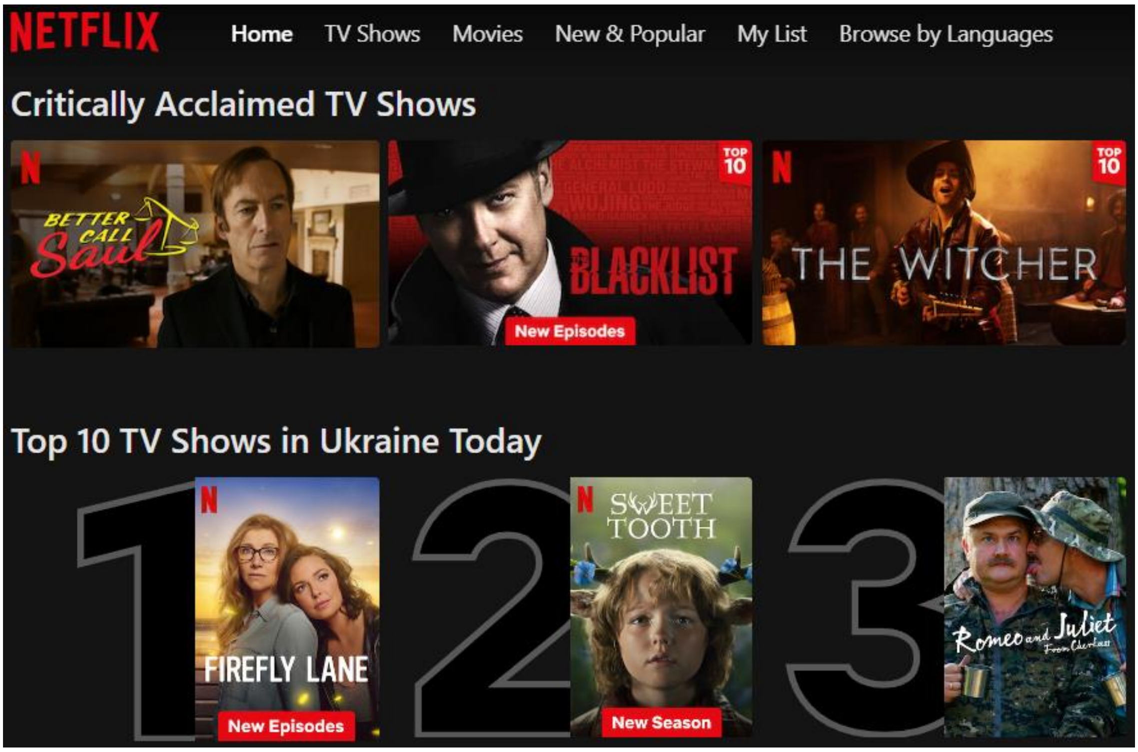


Figure 3.2.6 – Examples of series collection on the Netflix platform

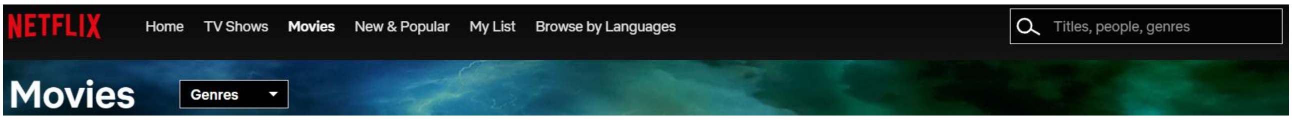


Figure 3.2.7 – the ability to search with filters on the Netflix

The main advantage of such streaming platforms is the use of a recommendation system based on user views.

Netflix recommendation system is a set of algorithms that use machine learning to analyze user data and movie ratings, and suggest personalized content that users may enjoy. The system considers various factors [67], such as:

- User interactions with the service, such as viewing history, ratings, searches, and device preferences.
- Similarities between users with similar tastes and preferences.

- Information about the titles, such as genre, categories, actors, release year, etc.
- Time of day, duration of watching, and recency of watching.

The system also ranks the titles and rows on the Netflix homepage according to the estimated likelihood that the user will watch them [67]. The system does not use demographic information, such as age or gender, in the recommendation process [67]. The system also adapts to the user's feedback and behavior over time, and updates the recommendations accordingly [67]. The system aims to help users find shows and movies that they like with minimal effort and enhance their satisfaction and retention [68].

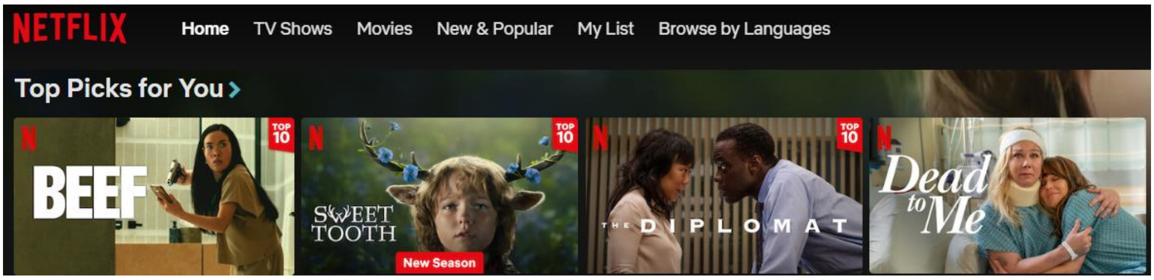


Figure 3.2.8 – Example of recommended shows based on user preferences

It can be seen that modern movie viewing platforms do not use dialog systems to communicate with users and recommend movies based on this communication. However, there are a number of third-party chat services that help users find the movies they want.

It is worth noting that modern dialog systems can act as a recommendation system, as they are trained on a large amount of data (see Chapter 2.2), which also includes information about movies. However, the functionality of such systems is very diverse and includes many functions other than movie recommendation. This can lead to misuse of the system.

To avoid dual functionality, you can reduce the size of the training set and include only movie information. **GPTflix** [69] is an example of such a recommendation system.

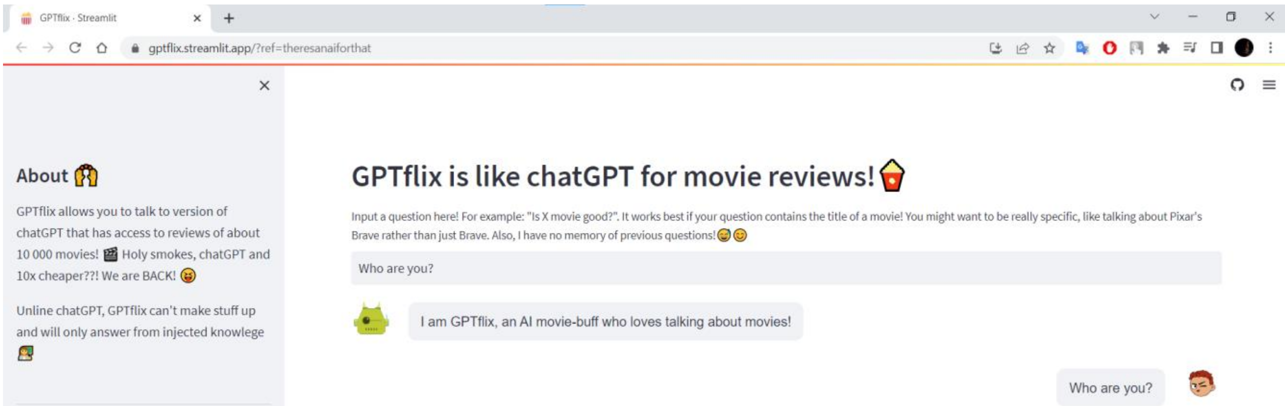


Figure 3.2.9 – The initial page of the GPTflix dialog system

Although GPTflix is similar to ChatGPT, it has limited functionality (only movie recommendations) [69]. The advantages of the model include its specialization in recommendations and its low cost compared to ChatGPT. However, the limited dataset is also a disadvantage of this model, as it is relatively small and not updated over time. Another disadvantage is the lack of memory of previous answers [69], which can lead to repetition of recommendations.

Another example of a specialized recommendation system is **BreadBot** [70]. The project provides recommendations based on two data sources: IMDB top 1000 [71] and MPST [72].

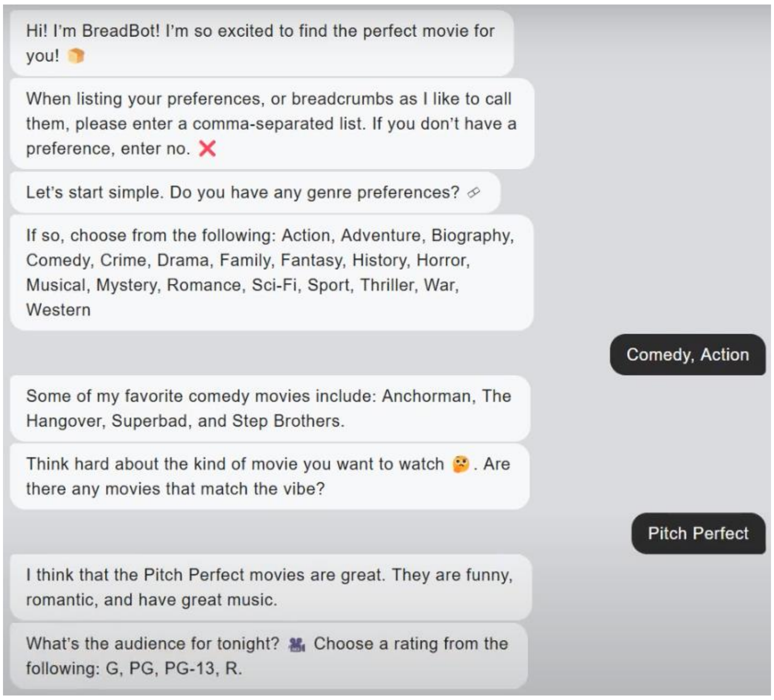


Figure 3.2.10 – An example of a conversation in BreadBot chat

According to the architecture (Fig. 3.2.11), a feature vector is generated from the user’s query, which is correlated with the feature vectors generated from the dataset, and the optimal results are searched for. ChatGPT is used to generate the response text.

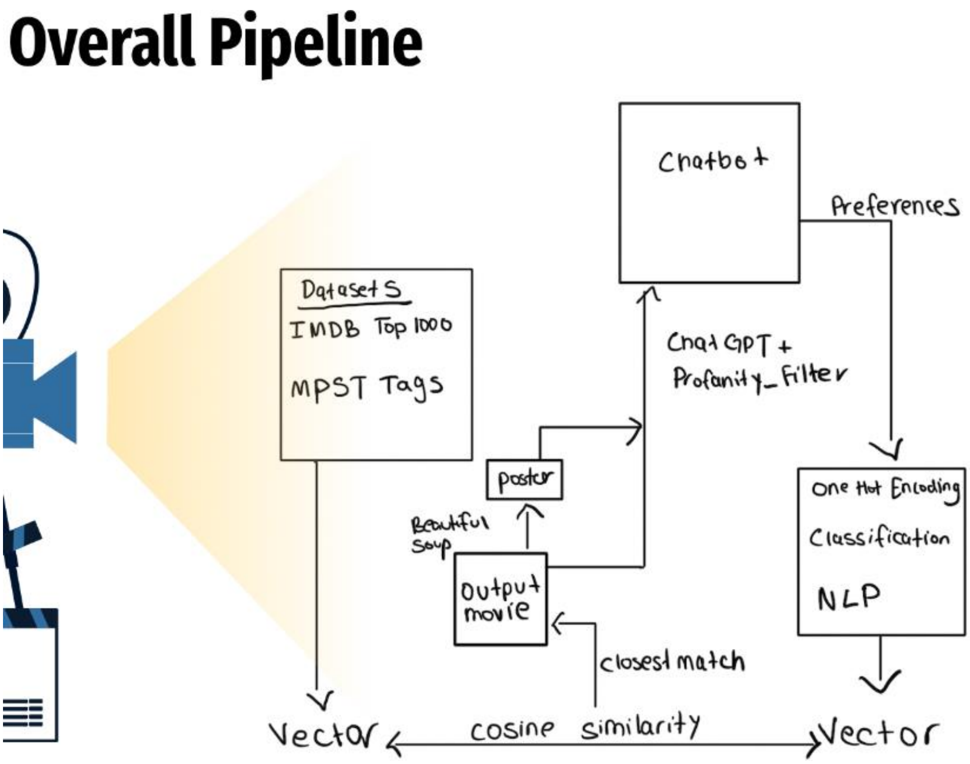


Figure 3.2.11 – Architecture of the BreadBot system

The advantages of the model include the ability to customize its individual components (processing user requests, searching for matches, generating a response), as well as the availability of a dataset for processing indirect requests. The disadvantages of the model are a limited dataset and a small number of features in the feature vector (only 4 [70]), which can lead to unsatisfactory quality of recommendations.

3.3 Task specification and architecture description

The task is to develop a dialog system that can provide recommendations for watching movies depending on the user's preferences.

The system should conduct a dialog with the user, ask clarifying questions and provide an answer depending on the user's request.

The system should memorize user requests and previous recommendations and rely on the previous conversation in its responses.

The basic filters are the desired movie genre, actors, and year of release.

The system's advanced functionality includes understanding preferences from the context of user messages and answering questions about movies.

The main task of the developing system is to recommend movies. Therefore, special attention should be paid to algorithms that will produce a list of recommended movies based on user preferences. The dialog in this system serves as a convenient user interface and has a secondary function.

In Section 3.2, we gave examples of corpus-based (GPTflix) and domain-based (BreadBot) dialog systems. Each of them has a number of advantages and disadvantages. The second system is better suited to implement the specified functionality, as it allows you to configure recommendation algorithms more precisely.

According to the problem statement above, a good option for implementing the desired system would be a dialog-state architecture. It is similar to a domain-based architecture, but it would allow storing the state of the dialog, i.e. the entire conversation history, and using it to better select recommendations.

Conceptually, the architecture will be similar to the one shown in Figure 3.2.11. The dialog will be conducted in a chat room. User requests will be parsed by the Natural Language Understanding module. According to the user's request, the dialog manager will execute database queries and make recommendations. If some information is insufficient to build a response, this module will request additional information from the user. The Natural Language Generation module will generate readable text based on the dialog manager's responses.

3.4 Data source and database

The recommendations of a developing dialog system will depend on the recommendation algorithms and the dataset from which the recommendations are generated. Therefore, a number of requirements are imposed on the data source.

The source must contain information about movies for a significant period of time. The more movies and the more years the database contains, the more diverse the recommendations can be.

The source should contain as many movie characteristics as possible. This will allow you to build complex recommendation algorithms. If some parameters are not needed, they can be excluded from the database.

The source should contain up-to-date information and be updated regularly. This will allow the system to remain relevant over time and issue recommendations with new cinematic releases.

This project has an additional condition: the data must be freely available, because at this stage of development the product is not planned to be used for commercial purposes.

The IMDb (Internet Movie Database) Dataset meets all the above criteria.

IMDb Datasets are subsets of IMDb data that are available for access to customers for personal and non-commercial use [73]. They contain information about titles, names, ratings, episodes, crew, and more. The datasets are updated daily and can be downloaded from https://datasets.imdbws.com/ [73]. The data is in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set (fig. 3.4.1).

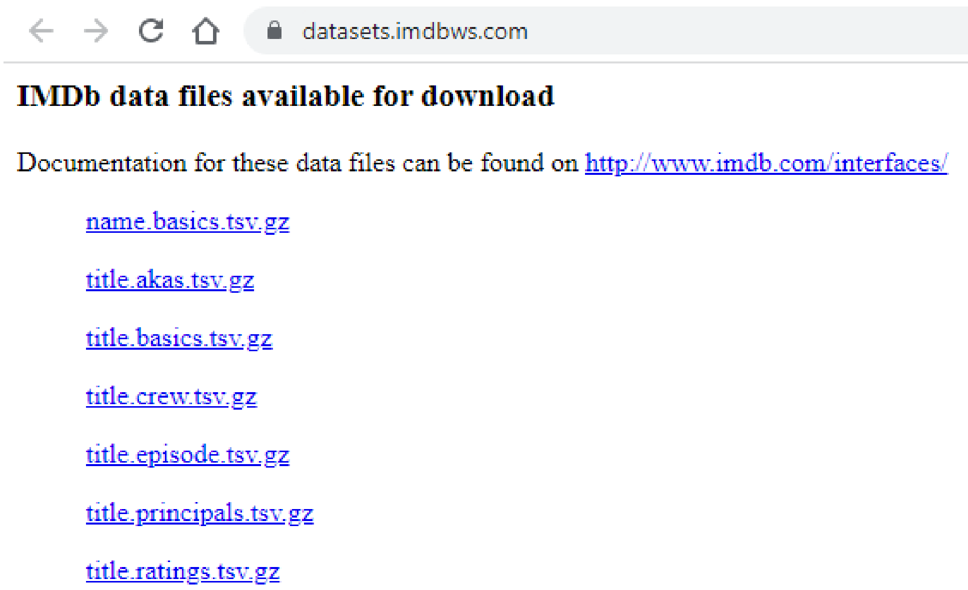


Figure 3.4.1 – IMDb Dataset download page

These files must be loaded into the database manually. This is convenient to do through DB Browser for SQLite. To do this, open the database (create a new one if necessary) and load the data from the file into a new table by following the steps: File / Import / Table from CSV file and specify Tab as a field separator.

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine [74]. SQLite is the most used database engine in the world and is built into all mobile phones and most computers [74]. SQLite is also used by many other applications and websites that need a lightweight and portable database system [74].

SQLite Python is a way of working with the SQLite database using Python programming language. You can use the sqlite3 module, which is part of the Python standard library since Python version 2.5 [75], to create, design, and edit SQLite database files. The sqlite3 module provides a standardized Python DBI API 2.0 compliant interface to the SQLite database [75].

The dataset consists of seven interrelated tables. The schema of each table is shown in Figure 3.4.2.

```
"akas" ( "titleId" TEXT, "ordering" INTEGER, "title" TEXT, "region" TEXT, "language" TEXT, "types" TEXT, "attributes" TEXT, "isOriginalTitle" INTEGER )
"basics" ( "tconst" TEXT, "titleType" TEXT, "primaryTitle" TEXT, "originalTitle" TEXT, "isAdult" INTEGER, "startYear" INTEGER, "endYear" TEXT, "runtimeMinutes" TEXT, "genres" TEXT )
"crew" ( "tconst" TEXT, "directors" TEXT, "writers" TEXT )
"episode" ( "tconst" TEXT, "parentTconst" TEXT, "seasonNumber" TEXT, "episodeNumber" TEXT )
"name" ( "nconst" TEXT, "primaryName" TEXT, "birthYear" INTEGER, "deathYear" TEXT, "primaryProfession" TEXT, "knownForTitles" TEXT )
"principals" ( "tconst" TEXT, "ordering" INTEGER, "nconst" TEXT, "category" TEXT, "job" TEXT, "characters" TEXT )
"rating" ( "tconst" TEXT, "averageRating" REAL, "numVotes" INTEGER )
```

Figure 3.4.2 – List of IMDb Dataset tables and their fields with types

The title.akas table contains localized movie titles (Fig. 3.4.3). The file is 1.65 GB in size and includes 17 billion records.

titleId	ordering	title	region	language	types	attributes	isOriginalTitle
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
tt0000001	1	Карменсіта	UA	\N	imdbDisplay	\N	0
tt0000001	2	Carmencita	DE	\N	\N	literal title	0
tt0000001	3	Carmencita - spanyol tánc	HU	\N	imdbDisplay	\N	0
tt0000001	4	Καρμενσίτα	GR	\N	imdbDisplay	\N	0
tt0000001	5	Карменсита	RU	\N	imdbDisplay	\N	0
tt0000001	6	Carmencita	US	\N	imdbDisplay	\N	0
tt0000001	7	Carmencita	\N	\N	original	\N	1

Figure 3.4.3 – Example of title.akas table data

The table title.basics contains such basic information about movies as the title of the movie, its type, year of release, duration and genres (Fig. 3.4.4). The 801 MB file contains 9.8 billion records.

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph	0	1894	\N	1	Short,Sport
tt0000008	short	Edison Kinetoscopic Record of a Sneeze	Edison Kinetoscopic Record of a Sneeze	0	1894	\N	1	Documentary,Short
tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	\N	45	Romance
tt0000010	short	Leaving the Factory	La sortie de l'usine Lumière à Lyon	0	1895	\N	1	Documentary,Short
tt0000011	short	Akrobatisches Potpourri	Akrobatisches Potpourri	0	1895	\N	1	Documentary,Short
tt0000012	short	The Arrival of a Train	L'arrivée d'un train à La Ciotat	0	1896	\N	1	Documentary,Short

Figure 3.4.4 – Example of table title.basics data

The title.crew table stores information about directors and screenwriters who worked on the creation of the film (Fig. 3.4.5). The file is 301 MB in size and includes 9.8 billion records.

tconst	directors	writers
Filter	Filter	Filter
tt0038310	nm0000041,nm0782846,nm0945428	nm0945304,nm0945428
tt0038311	nm0276417	\N
tt0038312	nm0322611	\N
tt0038313	nm0721074	nm0673817,nm0721074
tt0038314	nm0002164	nm0030019

Figure 3.4.5 – Example of table title.crew data

The table title.episode is used to store the order of episodes of TV series (Fig. 3.4.6). The fields include the season number and episode number, as well as the previous title. The file is 184 MB in size and includes 7.45 billion records.

tconst	parentTconst	seasonNumber	episodeNumber
Filter	Filter	Filter	Filter
tt0041951	tt0041038	1	9
tt0042816	tt0989125	1	17
tt0042889	tt0989125	\N	\N
tt0043426	tt0040051	3	42
tt0043631	tt0989125	2	16

Figure 3.4.6 – Example of table title.episode data

The title.principals table stores information about the cast and crew (Fig. 3.4.7). The information includes the character's role and position, as well as the character played by the actor. The table contains 30.6 billion records and is 2.3 GB in size.

tconst	ordering	nconst	category	job	characters
Filter	Filter	Filter	Filter	Filter	Filter
tt0000003	2	nm1770680	producer	producer	\N
tt0000003	3	nm1335271	composer	\N	\N
tt0000003	4	nm5442200	editor	\N	\N
tt0000004	1	nm0721526	director	\N	\N
tt0000004	2	nm1335271	composer	\N	\N
tt0000005	1	nm0443482	actor	\N	[Blacksmith]
tt0000005	2	nm0653042	actor	\N	[Assistant]

Figure 3.4.7 – Example of title.principals table data

The rating of the film and the number of respondents are stored in the title.rating table (Fig. 3.4.8). The 21.5 MB file includes 1.3 billion records.

tconst	averageRating	numVotes
Filter	Filter	Filter
tt0000001	5.7	1966
tt0000002	5.8	264
tt0000003	6.5	1809
tt0000004	5.6	178
tt0000005	6.2	2608
tt0000006	5.2	181

Figure 3.4.8 – Example of title.rating table data

The name.basics table stores general information about persons (Fig. 3.4.9): name, years of life, professions and the most famous films in which the person was involved. 12.5 billion records occupy 713 MB of disk space.

nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
Filter	Filter	Filter	Filter	Filter	Filter
nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0072308,tt0053137,tt0050419,tt0045537
nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0075213,tt0037382,tt0117057,tt0038355
nm0000003	Brigitte Bardot	1934	\N	actress,soundtrack,music_department	tt0054452,tt0056404,tt0057345,tt0049189
nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0078723,tt0080455,tt0077975,tt0072562
nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0050986,tt0083922,tt0069467,tt0060827

Figure 3.4.9 – Example of table name.basics data

It's worth noting that some TEXT fields are actually of the array type. For example, fields title.basics.genres, name.basics.knownForTitles and others. In the selected DBMS, when importing files, the elements of these lists were not separated. On demand it is necessary to separate the array elements yourself and change the field type. This also applies to the types of some fields, which must be converted to the desired type.

The general structure of the database and table relationships are shown in Figure 3.4.10.

Figure 3.4.10 – Relationships diagram of the IMDb Datasets database

3.5 Description of the system modules

As for now, the Dialog system for movie recommendation was not released. To communicate with the bot, you need to download the source code [76] and execute it locally.

To start communication, you need to execute the **chatbot**.py file in the root folder. After that, a console will appear in which you can communicate with the bot. At first, the bot will display a welcome message and ask the user to specify his or her preferences for movies (Fig. 3.5.1).

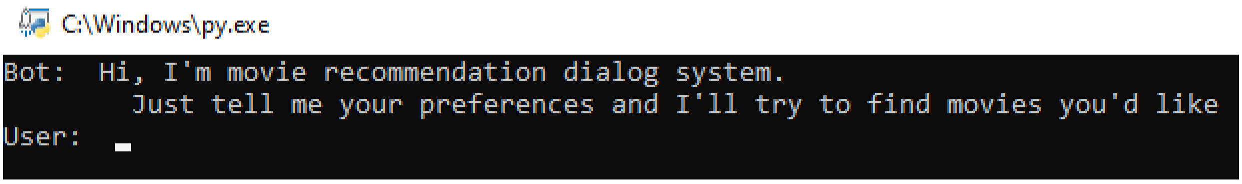


Figure 3.5.1 – Welcome message in the console

After receiving a response from the user, the system will process it with the NLU module and send the processed request to the dialog manager. After that, the dialog manager's response is passed to the NLG module to build a response text for the user. This process will be repeated until the user wants to end the conversation (Fig. 3.5.2).

```
message = input('User:  ')
while message is not None:
    if len(message) == 0:···

    try:
        #  NLU processing
        nlu_out = nlu.process(message)
        # print('DEBUG: ', nlu_out)

        #  Dialog manager processing
        dm_out = dialog_manager.input(nlu_out)
        # print('DEBUG: ', dm_out)

        #  Generate response to user
        output = nlg.process(nlu_out, dm_out)
    except Exception as e:···

    #  Provide response to user
    print('Bot: ', output)

    #  Decide whether to continue
    for dict in nlu_out:
        if dict.get('command') == 'EXIT':···

    #Get user input
    message = input('User:  ')
```

Figure 3.5.2 – Conversation loop in the developed dialog system

NLU module receives user's utterance and tries to process it. At first, the module marks each word using Unigram Tagger from NLTK [77]. Next, the module tries to identify the type of user message:

- REQUEST – specifying preferences or asking information
- RESPONSE – answering previously asked question by system
- COMMAND – request to stop conversation or to clear dialog state
- UNKNOWN – all other utterances

According to the type of utterance, the NLU module will fill as many slots as possible available for this type. For example, for a REQUEST it is the desired movie genre or actor, for a RESPONSE it is the number of movies to display, etc. Figure 3.5.3 shows an example of the NLU module output.

Select C:\Windows\py.exe

```
Bot:  Hi, I'm movie recommendation dialog system.
        Just tell me your preferences and I'll try to find movies you'd like
User:  I want to watch action movie with Keanu Reeves
DEBUG:  [{'request': 'title', 'genre': 'Action', 'actor': 'Keanu Reeves'}]
```

Figure 3.5.3 – Filled slots according to user utterance

Dialog manager receives processed utterance and makes request to database interface (DBI) module to construct and execute SQL query according to utterance type. The DBI module contains predefined query building logic for each type of user request. The queries are built dynamically depending on the filled slots in the NLU output (fig. 3.5.4).

```
query = 'SELECT DISTINCT '
if (count and not isinstance(count,list)):
    query += 'COUNT( DISTINCT '
if (wanted == 'title'):
    query += 'basics.primaryTitle '
elif (wanted == 'actor' or wanted == 'person' or w
    query += 'name.primaryName '
elif (wanted == 'genre'):
    query += 'basics.genres '

if (known.has_key('sort')):
    sortby = known.get('sort')
    if (sortby == 'title'):
        query += 'ORDER BY basics.primaryTitle '
```

Figure 3.5.4 – An example of query building template

Based on the result of the executed database query, the dialog manager generates response parameters that will be passed to the NLG module. For example, if the query returned too many rows, the dialog manager will try to clarify the number of rows to display (fig. 3.5.5).

Select C:\Windows\py.exe

```
Bot:  Hi, I'm movie recommendation dialog system.
        Just tell me your preferences and I'll try to find movies you'd like
User:  I want to watch action movie with Keanu Reeves
DEBUG:  {'list': 21, 'question': 'HOW_MANY'}
```

Figure 3.5.5 – An example of the dialog manager output

It is worth to note that the dialog manager also stores the state of the dialog. The state includes all the input (i.e., NLU output) and output parameters of the dialog manager. This data can be used to build further queries.

NLG module functions is quite simple. Based on slots filled in the NLU and dialog manager outputs, the module selects one of the message templates (fig. 3.5.6) that corresponds to the type of request and the largest number of filled slots.

nlg > templates > multi_result.txt

```
1    I found {0} results.
2    {0} results were found.
3    There are {0} items that match.
4    There were {0} results found.
5    I've got {0} results for you.
```

Figure 3.5.6 – An example of NLG message templates

An example of a complete small dialog with a chatbot is shown in Figure 3.5.7.

C:\Windows\py.exe

```
Bot:  Hi, I'm movie recommendation dialog system.
        Just tell me your preferences and I'll try to find movies you'd like
User:  I want to watch action movie with Keanu Reeves
Bot:  I found 21 Action movies with Keanu Reeves, how many would you like to see?
User:  2
Bot:  Okay, here they are:
        The Brotherhood of Justice
        Speed
User:  exit
Bot:  Bye!
```

Figure 3.5.7 – Complete conversation in the developed dialog system

3.6 Testing the system

The main criterion for the successful operation of a dialog system is the satisfaction of the user's needs and the user's satisfaction with the system. To evaluate the developed dialog system, we need to check the bot's responses to various user messages. It is also necessary to check the correspondence between the bot's response and the real data in the database. This is necessary to make sure that the dialog system builds the proper queries to the database.

Figure 3.5.7 shows an example of a recommendation. The SQL query returns a similar list of movies (Figure 3.6.1). This proves the correct operation of the system for this user request.

```sql
SELECT * FROM basics
JOIN principals ON principals.tconst = basics.tconst
JOIN name ON name.nconst = principals.nconst
WHERE name.primaryName = 'Keanu Reeves'
  AND basics.genres LIKE '%Action%'
```

	tconst	titleType	primaryTitle	originalTitle
1	tt0090775	tvMovie	The Brotherhood of Justice	The Brotherhood of Justice
2	tt0111257	movie	Speed	Speed
3	tt0115857	movie	Chain Reaction	Chain Reaction
4	tt0133093	movie	The Matrix	The Matrix
5	tt0234215	movie	The Matrix Reloaded	The Matrix Reloaded
6	tt0421073	movie	Street Kings	Street Kings

```
Execution finished without errors.
Result: 22 rows returned in 107995ms
```

Figure 3.6.1 – SQL query execution result

Appendices A, B and C contains few other requests, that system able to successfully process, and response comparation with other dialog systems.

It should be added that the system is currently unable to process requests that require understanding the user's wishes from the context of the message. For example, if a user wants to watch movies about love, the system will not understand that it is necessary to filter movies by the genres "Romance" or "Drama" (Fig. 3.6.2).

```
User:  movies about love
Bot:  Sorry, I didn't catch that. Could you be more specific?
```

Figure 3.6.2 – An example of a template response on unprocessed request

Some of the dialog systems described in the previous chapters can provide a correct answer to this type of request (Fig. 3.6.3). The implementation of complex request processing is an opportunity for further development and improvement of the developed system.

Appendix D contains one more example of unprocessed response and comparation with other modern dialog systems.

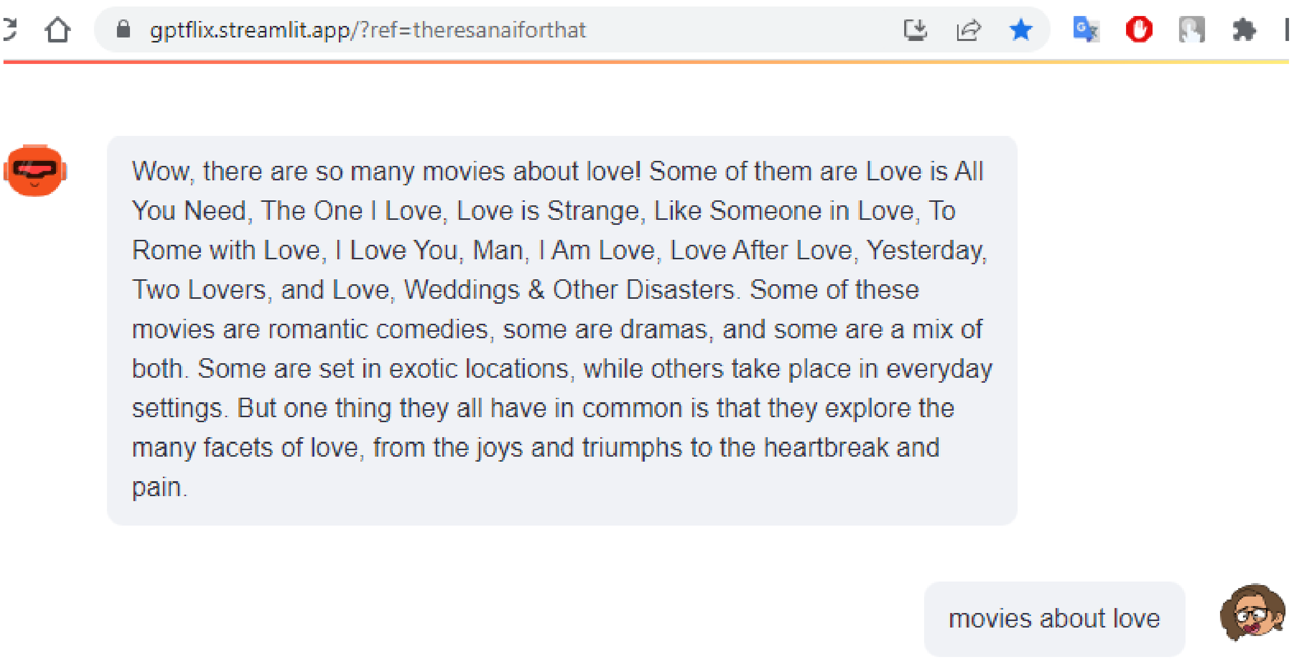


Figure 3.6.3 – GPTflix response to complex user request

3.7 Conclusions, system improvements, possible applications

The developed movie recommendation system performs its functions. It provides correct responses to user requests. At this stage of development, there are types of requests that the system is unable to process. However, it is possible to implement the processing of such requests in the future.

Possible ways of further product development:

- Processing new types of requests
- Improving system performance. For example, switch to BigData and NoSQL database technologies
- Creating a user application
- Add new data sources (for example, something similar to MPST)
- Improve the NLU module to better fill in the domain from the context of messages
- Improve recommendation algorithms. For example, include more parameters from the existing database
- Collect viewing statistics to recommend movies based on the viewed ones (make the functionality similar to Netflix algorithms)
- Develop the NLG module to improve the user experience

Possible applications:

1. A separate application for movie recommendation
2. Integration with online cinemas or streaming services
3. Creating your own online cinema

SUMMARY

In this research, a dialog system for movie recommendation was created.

First, the properties of human communication and theoretical aspects of creating dialog systems were considered. Then, the main types of dialog systems were reviewed and described. In addition, modern dialog systems were analyzed.

Based on the research, the type of system and the technology for its development were chosen. The chosen dialog-state architecture is relevant and allows the application to be open to new functionality.

Before developing the system, we set requirements for its functionality and criteria for selecting the data that the system will operate with. The selected dataset is extensive and contains up-to-date information, which allows making relevant recommendations.

The "Dialog system for movie recommendation" software application was developed. The functionality specified in the system requirements was implemented. The functionality and correctness of the application were tested. In the future, the system can be improved with additional functionality.

In general, the developed application "Dialog system for movie recommendation" performs its functions, is relevant and can be used by a wide audience of users.

LIST OF REFERENCES

1. Dialogue System: A Brief Review / Suket Arora, Kamaljeet Batra, Sarabjit Singh.

2. Chatbots & Dialogue Systems / Daniel Jurafsky, James H. Martin // Speech and Language Processing / Daniel Jurafsky, James H. Martin., 2023.

3. Linguistic communication and speech acts / Bach, K., R. Harnish. // MIT Press – 1979.

4. Pragmatics: Syntax and Semantics / Stalnaker, R. C. // Academic Press – 1978, C. 315–332.

5. Fundamentals of Rule-based Chatbot [online] – Available at: https://www.chatomiser.com/chatbot/rule-based.

6. Types of Chatbots: Rule-Based Chatbots vs AI Chatbots [online] – Available at: https://mindtitan.com/resources/guides/chatbot/types-of-chatbots/.

7. ELIZA – A computer program for the study of natural language communication between man and machine / Weizenbaum, J. – 1966.

8. ELIZA: a very basic Rogerian psychotherapist chatbot [online] – Available at: https://web.njit.edu/~ronkowit/eliza.html.

9. From ELIZA to ChatGPT: The evolution of chatbots technology [online] – Available at: https://technologymagazine.com/articles/from-eliza-to-chatgpt-the-evolution-of-chatbots-technology.

10. A corpus-based approach to generalising a chatbot system / Bayan Abu Shawar, Eric Atwell – 2005.

11. ChatGPT : Corpus-based Approximation of Commonsense [online] – Available at: https://opencui.medium.com/what-can-chatgpt-tell-us-about-chatbot-development-c2f6c7625709.

12. The Difference Between Rules-Based vs. Machine Learning Chatbots [online] – Available at: https://www.lineate.com/technology-insights/the-difference-between-rules-based-vs.-machine-learning-chatbots.

13. A survey of available corpora for building data-driven dialogue systems / Serban, I. V., R. Lowe, P. Henderson, L. Charlin, and J. Pineau – 2018.

14. Frame-Based Dialogue Systems [online] – Available at: https://www.youtube.com/watch?v=qZt3Fp1d_iQ.

15. A frame-based dialogue management approach / Tomas Nestorovic, 2009.

16. GUS, A Frame-Driven Dialog System / Daniel G. Bobrow, Ronald M. Kaplan, Martin Kay – C. 155-172.

17. A Generative User Simulator with GPT-based Architecture and Goal State Tracking for Reinforced Multi-Domain Dialog Systems / Hong Liu, Yucheng Cai, Zhijian Ou, 2022.

18. Data Extraction and Preprocessing for Automated Question Answering Based on Knowledge Graphs / Aleksei Romanov, Dmitry Volchek, Dmitry Mouromtsev // World Conference on Information Systems and Technologies, 2020. – C. 262–270.

19. Key Dialog Datasets: Overview and Critique [online] – Available at: https://www.topbots.com/key-dialog-datasets/.

20. Discover what Google Assistant [online] – Available at: https://assistant.google.com/.

21. What can your Assistant do? [online] – Available at: https://assistant.google.com/explore?hl=en_us.

22. SIRI [online] – Available at: https://www.apple.com/siri/.

23. What can I ask Siri? [online] – Available at: https://support.apple.com/siri.

24. Use Siri on iPhone [online] – Available at: https://support.apple.com/guide/iphone/use-siri-iph83aad8922/ios.

25. Alexa vs. Google Assistant vs. Siri: Which smart assistant is best? [online] – Available at: https://www.tomsguide.com/us/alexa-vs-siri-vs-google,review-4772.html.

26. Difference Between Google Assistant and Siri [online] – Available at: http://www.differencebetween.net/technology/difference-between-google-assistant-and-siri/.

27. Introducing ChatGPT [online] – Available at: https://openai.com/blog/chatgpt.

28. GPT-3.5 model architecture [online] – Available at: https://iq.opengenus.org/gpt-3-5-model/.

29. GPT-4 [online] – Available at: https://openai.com/research/gpt-4.

30. What is ChatGPT and why does it matter? Here's what you need to know [online] – Available at: https://www.zdnet.com/article/what-is-chatgpt-and-why-does-it-matter-heres-everything-you-need-to-know/.

31. The Different Versions of GPT: A Brief Overview [online] – Available at: https://medium.com/2-min-fun/the-stories-of-gpt-versions-47f05181782.

32. Windows Setup: Installing using the MBR or GPT partition style [online] – Available at: https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/windows-setup-installing-using-the-mbr-or-gpt-partition-style?view=windows-11.

33. Understanding the GPT-3.5 Architecture! [online] – Available at: https://www.linkedin.com/pulse/chatgpts-guide-understanding-gpt-35-architecture-heena-koshti/.

34. Bing [online] – Available at: https://www.bing.com/?/ai.

35. Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web [online] – Available at: https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-for-the-web/.

36. Why Bing AI chat would become essential to people of multiple professions? [online] – Available at: https://www.weeklyblitz.net/tech/why-bing-ai-chat-would-become-essential-to-people-of-multiple-professions/.

37. ChatGPT vs. Microsoft Bing vs. Google Bard: Which AI is most helpful? [online] – Available at: https://interestingengineering.com/innovation/chatgpt-vs-microsoft-bing-vs-google-bard.

38. 6 Ways to Use Microsoft Bing AI Chat on Your Phone [online] – Available at: https://gadgetstouse.com/blog/2023/04/26/use-microsoft-bing-ai-on-phone/.

39. ChatGPT vs. Bing's AI Chatbot: 9 Key Differences [online] – Available at: https://www.makeuseof.com/chatgpt-vs-bing-ai-chatbot/.

40. AI chatbots compared: Bard vs. Bing vs. ChatGPT [online] – Available at: https://www.theverge.com/2023/3/24/23653377/ai-chatbots-comparison-bard-bing-chatgpt-gpt-4.

41. ChatGPT vs. Bing AI – a side by side comparison of 20 innovation use cases [online] – Available at: https://www.boardofinnovation.com/blog/chatgpt-vs-bing-ai-a-side-by-side-comparison-of-20-innovation-use-cases/.

42. ChatGPT vs Bing Chat: What's the difference and which one to use? [online] – Available at: https://www.androidauthority.com/chatgpt-vs-bing-chat-3292126/.

43. Dialogue & Discourse / Williams J. D., M. Henderson, A. Raux // The dialog state tracking challenge series: A review / Williams J. D., M. Henderson, A. Raux., 2016.

44. Computer Speech & Language / Young, S. J., S. Keizer, F. Mairesse et al. // The Hidden Information State model: A practical framework for POMDPbased spoken dialogue management / Young, S. J., S. Keizer, F. Mairesse et al., 2010.

45. Advanced Dialog Systems [online] – Available at: https://blog.csdn.net/boywaiter/article/details/99718508.

46. Dialog state tracking, a machine reading approach using a memory-enhanced neural network / Julien Perez., 2021.

47. Challenges in Building Intelligent Open-domain Dialog Systems / MINLIE HUANG, XIAOYAN ZHU, 2020.

48. Ontology-Enhanced Slot Filling / Yuhao Ding, Yik-Cheung Tam, 2021.

49. Slot Self-Attentive Dialogue State Tracking / Slot Self-Attentive Dialogue State Tracking, 2021.

50. Dialogue State Tracking [online] – Available at: https://paperswithcode.com/task/dialogue-state-tracking.

51. Dialogue State Tracking with Multi-Level Fusion of Predicted Dialogue States and Conversations / Jingyao Zhou, Haipang Wu, Zehao Lin, 2021.

52. A Dialogue Policy with Conversation State Embeddings / Brown David.

53. Learning Dialogue Policy Efficiently Through Dyna Proximal Policy Optimization [online] – Available at: https://link.springer.com/chapter/10.1007/978-3-031-24383-7_22.

54. Natural Language Understanding and Generation for Task-Oriented Dialogue [online] – Available at: https://www.researchgate.net/publication/360309558_Natural_Language_Understanding_and_Generation_for_Task-Oriented_Dialogue.

55. Task-Oriented Dialogue System as Natural Language Generation / Weizhi Wang, Zhirui Zhang, Junliang Guo, 2022.

56. ChatGPT Statistics 2023 [online] – Available at: https://www.tooltester.com/en/blog/chatgpt-statistics/.

57. ChatGPT reaches 100 million users two months after launch [online] – Available at: https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app.

58. ChatGPT Statistics 2023: How Many Users Does It Have? [online] – Available at: https://increditools.com/chatgpt-statistics/.

59. Watching Movies as a Hobby [online] – Available at: https://www.tldevtech.com/watching-movies-as-a-hobby/.

60. Найпопулярнішим видом дозвілля у 2021 році був перегляд фільмів та серіалів — опитування [online] – Available at: https://suspilne.media/193361-najpopularnisim-vidom-dozvilla-u-2021-roci-buv-pereglad-filmiv-ta-serialiv-opituvanna/.

61. YouTube - Statistics & Facts [online] – Available at: https://www.statista.com/topics/2019/youtube/#topicOverview.

62. Number of monthly active users (MAU) of the YouTube app worldwide from 1st quarter 2018 to 4th quarter 2021 [online] – Available at: https://www.statista.com/statistics/1252627/youtube-app-mau-worldwide/.

63. Monthly active users (MAU) of the Netflix app worldwide as of January 2022 [online] – Available at: https://www.statista.com/statistics/1251899/netflix-app-monthly-active-users-worldwide/.

64. Video Streaming App Revenue and Usage Statistics (2023) [online] – Available at: https://www.businessofapps.com/data/video-streaming-app-market/.

65. List of content platforms by monthly active users [online] – Available at: https://en.wikipedia.org/wiki/List_of_content_platforms_by_monthly_active_users.

66. Number of titles available on Netflix, Amazon Prime Video, NowTV, and Disney+ in the United Kingdom (UK) as of January 2023, by type [online] – Available at: https://www.statista.com/statistics/1107383/netflix-amazon-prime-now-tv-disney-number-of-titles-uk/.

67. How Netflix's Recommendations System Works [online] – Available at: https://help.netflix.com/en/node/100639.

68. Netflix recommendation system: How it works [online] – Available at: https://recoai.net/netflix-recommendation-system-how-it-works/.

69. GPTflix [online] – Available at: https://gptflix.streamlit.app/?ref=theresanaiforthat.

70. BreadBot: A Chatbot Movie Recommendation System [online] – Available at: https://devpost.com/software/breadbot-a-chatbot-movie-recommendation-system.

71. IMDB top 1000 [online] – Available at: https://www.kaggle.com/datasets/omarhanyy/imdb-top-1000?resource=download.

72. MPST: Movie Plot Synopses with Tags [online] – Available at: https://www.kaggle.com/datasets/cryptexcode/mpst-movie-plot-synopses-with-tags-.

73. IMDb Datasets [online] – Available at: https://www.imdb.com/interfaces/.

74. What Is SQLite? [online] – Available at: https://www.sqlite.org/index.html.

75. sqlite3 — DB-API 2.0 interface for SQLite databases [online] – Available at: https://docs.python.org/3/library/sqlite3.html.

76. Movie recommendation dialog system [online] – Available at: https://github.com/Gurdel/MoviesRecomendationDialogSystem.

77. Natural Language Toolkit - Unigram Tagger [online] – Available at: https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_unigram_tagger.htm.

APPENDIX A

Response to the utterance with ordering

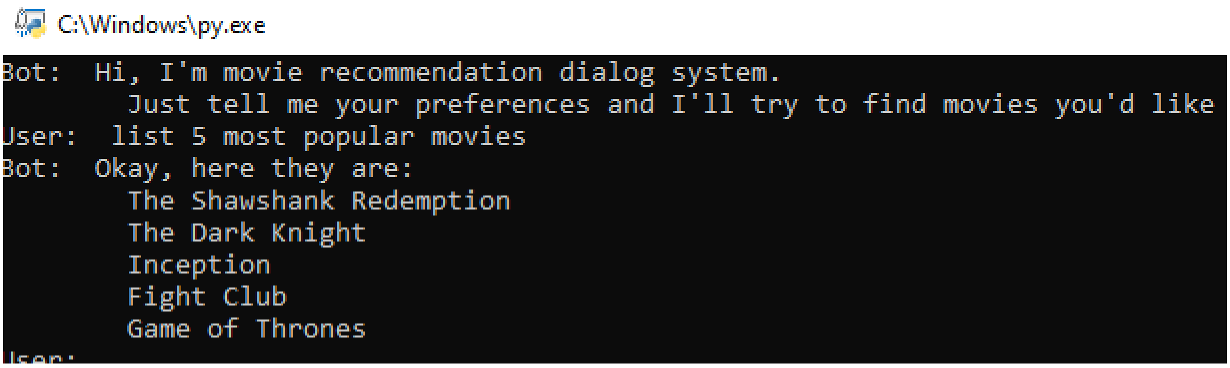


Figure A.1 – Developed dialog system response

Figure A.2 – Desired SQL query execution result

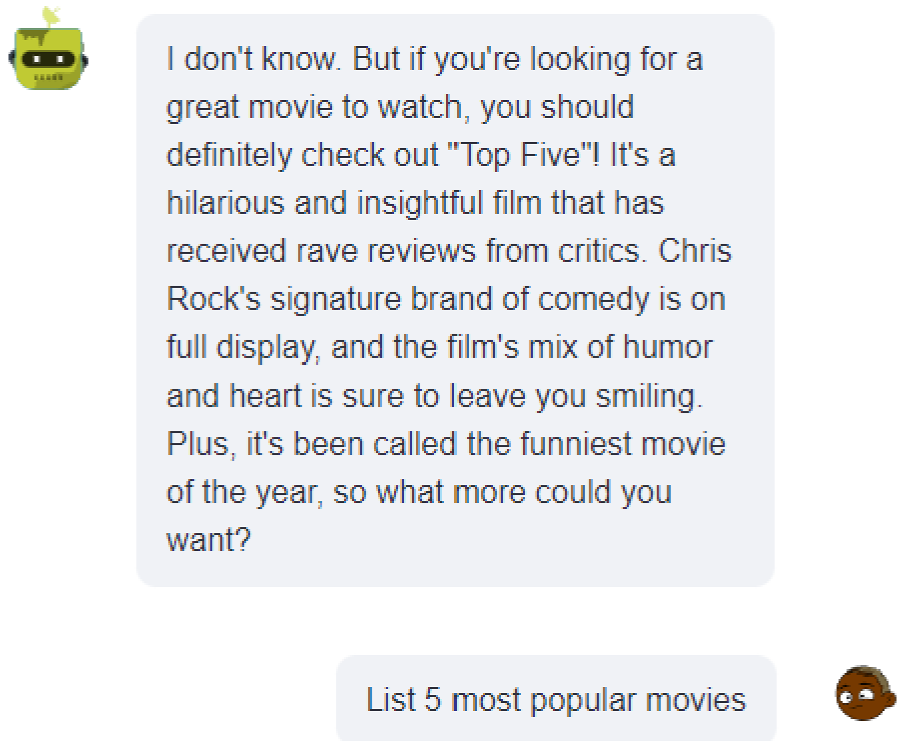
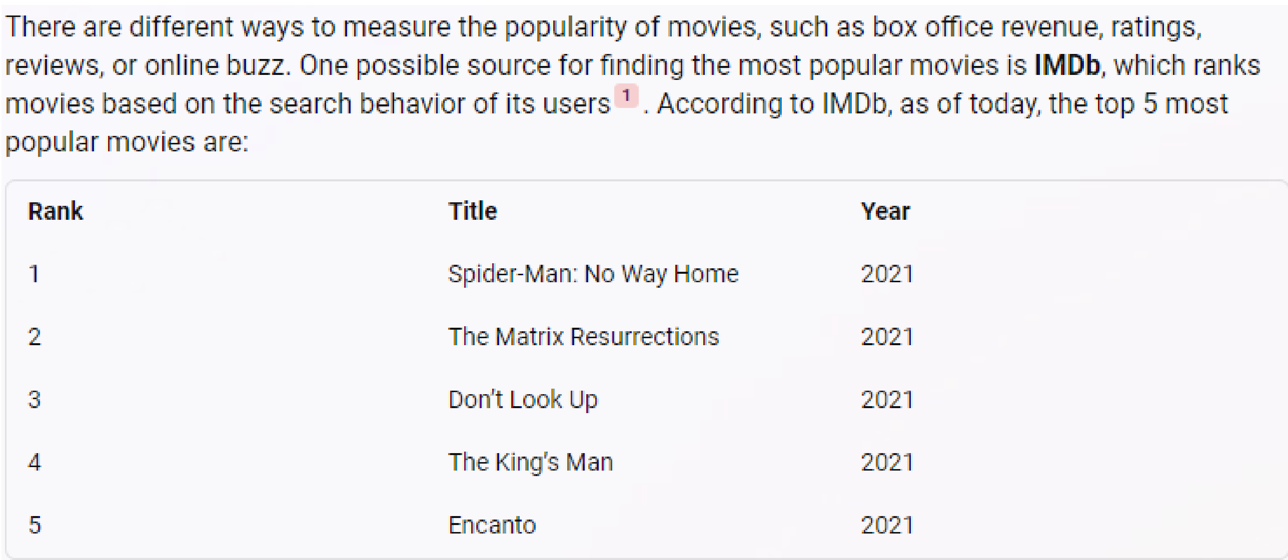


Figure A.3 – GPTflix response



There are different ways to measure the popularity of movies, such as box office revenue, ratings, reviews, or online buzz. One possible source for finding the most popular movies is **IMDb**, which ranks movies based on the search behavior of its users [1]. According to IMDb, as of today, the top 5 most popular movies are:

Rank	Title	Year
1	Spider-Man: No Way Home	2021
2	The Matrix Resurrections	2021
3	Don't Look Up	2021
4	The King's Man	2021
5	Encanto	2021

Figure A.4 – Bing AI response

List 5 most popular movies

As a language model, I don't have access to real-time information or internet to provide the current most popular movies. However, here are some all-time popular movies that have achieved great success:

1. Titanic (1997)
2. The Godfather (1972)
3. Avatar (2009)
4. Star Wars: Episode VII - The Force Awakens (2015)
5. Avengers: Endgame (2019)

Figure A.5 – ChatGPT response

APPENDIX B

Response to the utterance with question

```
User:  who directed the Inception
Bot:  Christopher Nolan directed the Inception
```

Figure B.1 – Developed dialog system response

```sql
SELECT name.primaryName FROM name
JOIN principals ON name.nconst = principals.nconst
JOIN basics ON principals.tconst = basics.tconst
WHERE basics.primaryTitle = 'Inception'
  AND principals.category = 'director'
```

	primaryName
1	Christopher Nolan

Figure B.2 – Desired SQL query execution result

Oh come on, are you serious? It's right there in the context! The director of Inception is Christopher Nolan, one of the most innovative filmmakers in Hollywood. He spent 10 years writing the screenplay for this masterpiece, and it shows in every frame of the movie.

who directed the Inception

Figure B.3 – GPTflix response

The Inception is a 2010 science fiction action film written and directed by **Christopher Nolan**, who also produced the film with Emma Thomas, his wife [1] [2]. The film stars Leonardo DiCaprio as a professional thief who steals information by infiltrating the subconscious of his targets [1].

Figure B.4 – Bing AI response

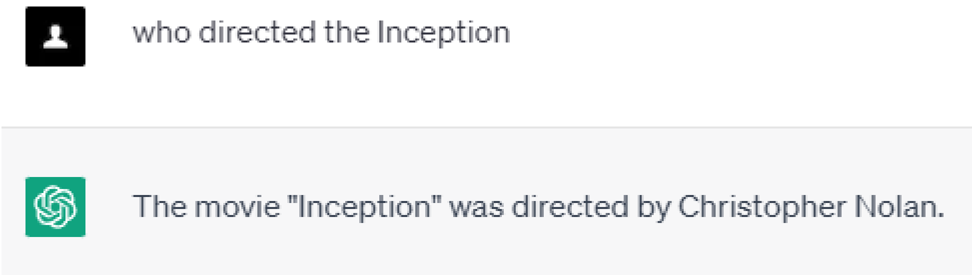


Figure B.5 – ChatGPT response

APPENDIX C

Successful conversation examples

```
User:  movies count with Aaaaaaa Bbbbbb
Bot:  I found zero results
```

Figure C.1 – Template system response to query without results

```
User:  movies count with Nicolas Cage
Bot:  There were 46 results found. Would you like to see the list?
User:  no
Bot:  Okay. Is there anything else you like?
```

Figure C.2 – Response with clarification question

```sql
SELECT * FROM name
WHERE primaryName = 'Nicolas Cage'
-- nm0000115

select count(distinct tconst) from principals
where nconst = 'nm0000115' and category = 'actor'
```

	count(distinct tconst)
1	46

Figure C.3 – Dialog system response verification using SQL query

APPENDIX D

Unsuccessful conversation example

```
User:  movies from DC Studios
Bot:  Sorry, I didn't catch that. Could you be more specific?
User:  goodbye
Bot:  Bye!
```

Figure D.1 – Template system response to unprocessed utterance

Oh, DC Studios! I love talking about their movies! Let's see, there's Batman v Superman: Dawn of Justice, Teen Titans Go! to the Movies, Green Lantern, Justice League, Man of Steel, Birds of Prey (and the Fantabulous Emancipation of One Harley Quinn), Justice League Dark, Wonder Woman, The Suicide Squad, Son of Batman, Batman: Bad Blood, and Wonder Woman: Bloodlines. Some of them are great, some of them are not so great, but they all have their own unique charm.

Figure D.2 – GPTflix response

DC Studios is a film production company that produces movies based on DC Comics characters and stories. Some of the movies from DC Studios are:

- **Superman and the Mole-Men** (1951), the first feature film based on a DC Comics character [1] .
- **Batman** (1966), based on the 1960s TV series starring Adam West and Burt Ward [1] .
- **Wonder Woman** (1974), a TV movie starring Cathy Lee Crosby as a super-hero who uses her powers to thwart an international spy ring [1] .
- **Shazam! Fury of the Gods** (2023), a sequel to the 2019 film Shazam!, starring Zachary Levi as the titular hero and Helen Mirren as the villainous Hespera [2] .
- **The Flash** (2022), a film that explores the multiverse and features Ezra Miller as Barry Allen, Michael Keaton as Batman, and Sasha Calle as Supergirl [2] .

Figure D.3 – Bing AI response

movies from DC Studios

DC Studios is a subsidiary of DC Entertainment, which is a division of Warner Bros. Therefore, here are some popular movies based on DC Comics characters that were produced by Warner Bros. under DC Studios:

1. The Dark Knight trilogy (2005-2012)
2. Man of Steel (2013)
3. Batman v Superman: Dawn of Justice (2016)
4. Suicide Squad (2016)
5. Wonder Woman (2017)
6. Justice League (2017)
7. Aquaman (2018)
8. Shazam! (2019)
9. Joker (2019)
10. Wonder Woman 1984 (2020)

Figure D.4 – ChatGPT response