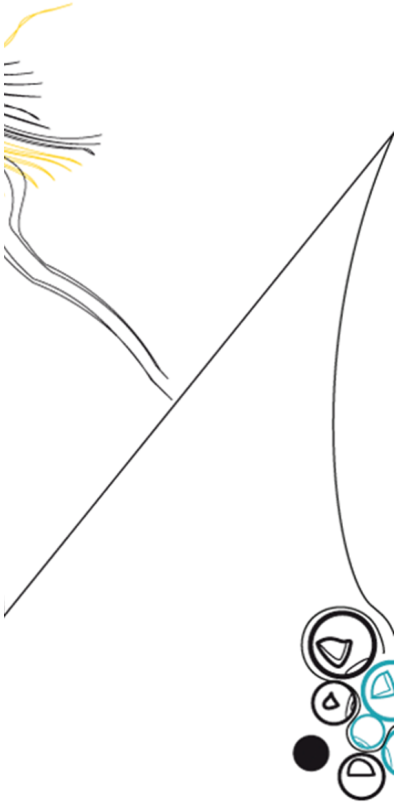
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Knowledge Graph Driven Conversational Virtual Museum Guide

Dou Liu

M.Sc. Thesis in Computer Science
December 20th 2021



Supervisors:
Dr. Mariët Theune
Dr. Shenghui Wang
Dr. João Luiz Rebelo Moreira

Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

Abstract

Conversational agent in museum is studied and developed during the past 20 years, but it is still underexplored for museum conversational agents to utilize knowledge graph to guide the visitors and share knowledge with them. In this thesis, we addressed how to create a knowledge graph driven conversational virtual museum guide, as well as how to utilize the knowledge graph to develop its skills, such as Questions and Answers (Q&A) and recommendation, aiming to share information with the visitors when they are visiting a virtual exhibition and help them explore the knowledge graph through dialogue.

To address the aforementioned questions, the solution is composed of following aspects. A culture heritage knowledge graph, containing three thousand paintings and other entities related to the paintings, is designed and built by extracting relevant entities and relationships from Wikidata. Then, we constructed our conversational virtual museum guide with Dialogflow, where six functions are designed to support different kinds of tasks and conversations. Skills of the agent, such as Q&A and recommendation, are implemented as webhook service, i.e. backend service. The Q&A utilized a template-based method to identify the question types, generate the query of the question and fill the answer fetched in the slot of response template. The recommendation give suggested entities in the knowledge graph by combining the user's interest and similarity scores calculated using node embedding. Finally, our knowledge graph driven museum conversational guide is integrated with a virtual museum exhibition and can be visited by the public.

As for the evaluation, an online user study was carried out with 10 participants, which included two stages. The first was playing through the virtual museum and finishing given tasks with the conversational virtual museum guide, and the second stage was completing a questionnaire about the experience with the conversational guide. The questions related to the usability and functionality of the conversational agent are designed to measure its performance, which helps to validate if the conversational agent can have natural conversation and share knowledge as a museum guide. The result shows that most users agree that having conversations improves their experience, and they are satisfied with the functions of the agent, especially the recommendation.

Our research shows the possibility to integrate knowledge graph with museum conversational agent. Features driven by knowledge graph can improve the experience of visitors.

Keywords: Knowledge Graph, conversational agent, cultural heritage, Q&A, recommendation

List of Figures

List of Tables

Acronyms

CA Conversational Agent

CH Cultural Heritage

DM Dialogue Management

FastRP Fast Random Projection

KBQA Knowledge Based Question Answering

KG Knowledge Graph

LOD Linked Open Data

NLU Natural Language Understanding

Q&A Questions and Answers

RDF Resource Description Framework

SQL Structured Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

1 Introduction

For centuries, visiting a cultural site with a guide who is familiar with the subject has been a valuable and rewarding experience, from which you can not only see the exhibits with your own eyes, but also learn the fascinating stories and knowledge about them. With the digitalization of Cultural Heritage (CH) institutions such as museums and libraries, it is now possible to preserve cultural relics in digital form and to visit them online. To be specific, thanks to the Linked Open Data (LOD) principles and Knowledge Graph (KG) technologies, the rich knowledge from CH collections has been transformed into a form that is shareable, extensible and easily reusable. The online access to this information makes it possible to build a conversational virtual museum guide based on this knowledge. Compared to directly viewing or retrieving the information, a Conversational Agent (CA), which can have conversations with visitors, understand their utterances and respond in natural language, if done well, can enhance the user's experience when visiting a museum [1]. What's more, it can help people who are unable to visit cultural sites in person not miss the experience, especially in the period of pandemic, when museums are not open to the public.

1.1 Virtual Museum Exhibition

In a collaboration between the University of Twente and Rembrandthuis[1], a virtual museum with themed exhibition *HERE. Black in Rembrandt's time* (or *HIER. Zwart in Rembrandts tijd* in Dutch), see Figure 1, is built in Unity by Claudia Alessandra Libbi[2], a Master student of Interactive Technology, University of Twente. Within this virtual exhibition, a user can walk around and visit freely. The virtual exhibition consists of ten portrait paintings from Rembrandt's time, whose object of the portraits are black people, two paintings depicting scenes from the work and daily life of black people, as well as four paintings created by contemporary black artists.

In total, there are 16 neglected works of art featuring black people. This exhibition aims to reveal the image of black people in Rembrandt's time, which is not the stereotype of the secondary figures, but as the main object of the painting and depicted with dignity and respect. Rembrandt House Museum provides basic information as well as background stories for all 16 paintings in the virtual museum. In this thesis, our CA will be integrated into this virtual museum.

Figure 1: The exhibition *HIER. Zwart in Rembrandts tijd* in the virtual museum.

[1] https://www.rembrandthuis.nl/en/
[2] https://www.focusonemotions.nl/175-people-students/632-claudia-libbi

1.2 Challenges and Research Questions

Based on the existing background and conditions, the main goal of this dissertation is to build a knowledge graph driven conversational guide, and the guide can be embedded in the virtual museum to share information with the user when needed. To achieve this goal, it is required to develop a CH knowledge graph. Though there are some LOD data sources available to the public, CH knowledge graphs in English are rare and of no use unless being integrated in a domain-specific application. During the museum visit, the conversational agent should give the necessary information about the exhibits and extra knowledge that might be interesting to the visitor in a *non-intrusive* way. It means that the virtual guide should not disturb or spoil the visitor's experience, but share the knowledge with the visitor naturally. In an online environment, a museum visitor may start the visit with a painting, and then shows interest in the creator or the history of the work. Similarly, within the knowledge graph, from a starting point, the visitor could follow the relations in the knowledge graph to discover what is interesting to them. The main challenges are:

- *C1*: the CA needs a cultural heritage knowledge graph database to support its functions;
- *C2*: the CA should be designed to have natural interaction in non-intrusive way as the virtual museum guide;
- *C3*: the CA should be able to make the huge amount of information/knowledge accessible to visitors.

To address the challenges above, in this task, following research questions are going to be explored and answered:

- *RQ1*: how to collect a cultural heritage knowledge graph for conversational virtual museum agent.
- *RQ2*: how to build a knowledge graph driven conversational virtual museum guide which could have natural dialogue with visitors.
- *RQ3*: how should we share knowledge with visitors through our conversational agent for user's interest while being non-intrusive?

1.3 Thesis Structure

The structure of the rest of this thesis is presented below:

Chapter 2 describes the background knowledge and preliminaries of this article, namely knowledge based conversational question answering and recommendations in CA, the framework for developing conversational agents and knowledge graphs, such as graph structure, query language, graph database and data source. Chapter 3 gives a brief introduction to the relevant work about CA for museums. Chapter 4 introduces the design for building a knowledge driven conversational virtual museum guide. Chapter 5 is about user evaluation. And the last one, Chapter 6 concludes the approaches and discuss the main lessons learned, limitations and future work for our CA.

2 Background Knowledge and Preliminaries

This section presents background knowledge that is necessary to understand basic concepts of the knowledge graph driven conversational agent in this thesis. A conversational agent is a dialogue system which can conduct conversations with human via auditory or textual methods. And a knowledge graph describes real-world entities and their relationships through the structure of graphs, which is an effective and flexible to represent a knowledge base. For a CA, a KG provides a vast knowledge as well as the interconnection between entities, however, the research on museum conversational agent with knowledge graph driven features is rare. To have a better understanding of our thesis, in Section 2.1, we introduce two popular frameworks for building a conversational agent and give a comparison between them. Section 2.2 presents some preliminary work for our thesis in KG, including the choices of knowledge graph database and source. Then, Section 2.3 and Section 2.4 introduces technologies of knowledge graph driven question answering and recommendation in conversational agent application respectively.

2.1 Frameworks for Conversational Agent

There are lots of frameworks and open source projects which help developers to build a conversational agent's interface with convenience. Two of the most widely used chatbot platforms are considered in this project: **Rasa** and **Dialogflow**[3].

Rasa is an open source machine learning framework for building text-based and voice-based chatbots. And Dialogflow is a closed source Natural Language Understanding (NLU) platform with a fully functional API and graphical web interface. It combines machine learning and rule-based methods and supports multiple forms of inputs and outputs, including text and speech.

Rasa has two components, NLU and Core. To develop a chatbot with various functions, both Rasa NLU and Rasa Core are required to use. Rasa NLU deals with intents and entities, while Rasa Core handles fulfillment, which is the response strategy to user's utterance, and dialogue. As an open source python library, Rasa allows the developer to customize the bot in lower level using python. It is completely free to configure the NLU, Core, integration and deployment. But Rasa does not offer the hosting or deployment.

Dialogflow has complete tools for building a chatbot. Specifically, it is able to deal with the classification of intents and entities, the context and customized webhook. With the robust APIs of Dialogflow, we can define the entities and intents either via API or on the web interface. Besides that, the data of Dialogflow is hosted on the Google Cloud and easy to deploy on the application. Dialogflow offers webhooks, which allows developers to fulfill the response via custom script. By creating a webhook on Dialogflow, it allows the agent to fetch responses or retrieve items from the API or our own code.

Based on the above comparison, for the convenience of integration and deployment, we decide to use Dialogflow as the platform to develop our virtual conversational agent since the agent may need to be deployed in a VR museum and Dialogflow supports easy operations for deployment in Unity.

Dialogue Management for Conversational Agent As the coordinator and strategist of a conversational agent, Dialogue Management (DM) affects the quality of the dialogue and the satisfaction of the user. It is responsible for managing agent's components, such as intents identification and response strategy, as well as utterances history. By doing so, it can help the agent understand what the user says and decide what to do next.

Two approaches are often utilized for building a DM, handcrafted and statistical approaches [2].

For handcrafted approaches, such as finite state-based and frame-based DM, they are simple and widely used in industry. On the other hand, the handcrafting makes it time-consuming and painstaking to come up with the possible rules and heuristics. As a result, handcrafted approaches are suitable for small or domain-specific projects rather than open-domain chatbots, since the former require much fewer strategies.

It could be challenging to design a good strategy. For different applications, the strategies vary a lot. And the diversity of users makes it even harder for DM to capture the intents accurately and give appropriate actions. To cope with this, the statistical approaches use machine learning methods, so

[3]https://cloud.google.com/dialogflow

that the DM can learn the optimal strategies from the dialogue data. The performance of the training for DM relies on the quality and quantity of the corpora. And we need to design the possible flow of conversation manually in advance. In case of our thesis, the virtual museum guide is domain specific, which means the range of the topics for dialogue is much less than open domain CA, and we can focus on the topics that could happen in the museum. Thus, the DM of Dialogflow should be sufficient, and both machine learning and rule-based methods provided by Dialogflow will be used.

2.2 Knowledge Graphs

2.2.1 RDF

Resource Description Framework (RDF) is widely used as a format to store the data and structure of a knowledge graph. As we can get the RDF data and store them in our server as the knowledge graph database, we start this section with an introduction of RDF.

The success of the World Wide Web shows the power of using standardized mechanisms for information exchange and communication. HTML is the standard language for editable web pages and is used to convey information about the structure of human-oriented documents. For the Semantic Web, whose needs are even richer, RDF provides just such a flexible and domain-independent data model. Its basic building block is expressed as the form subject–predicate–object, also called a triple, as Figure 2 shows.

Similar to Entity Attribute Value model, in a triple, the subject represents an entity, the predicate represents features or aspects of the entity, expressing the relationship between subject and object, and the object is the value. And a triple has a unique identifier, called the Uniform Resource Identifier (URI) in RDF.

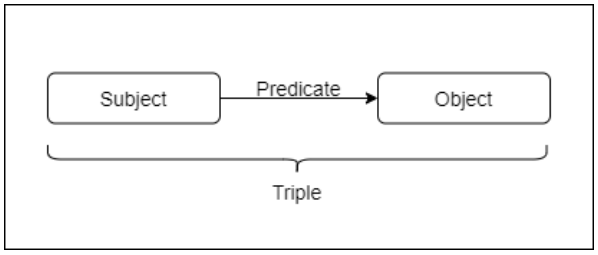


Figure 2: Triple: subject–predicate–object.

For example, one representation of the RDF concept "Rembrandt created the Night Watch" is a triple: a subject indicating "Rembrandt", a predicate indicating "create a work", and an object indicating "the Night Watch". In Wikidata, there are millions of triples like this to form the RDF dataset.

2.2.2 Graph Database

To save RDF data, a graph database is required. A graph database intends to preserve data without limiting it to a predefined data model. Relationships between data are treated as equally important to the data itself, as the LOD is not isolated, but interconnected. Only databases that support relationships themselves can store, process, and query effectively. While other databases compute relationships through expensive *JOIN* operations at query time, graph databases store the relationships in the model along with the data. In this way, it can show how each individual entity is associated with or related to another entity.

Regardless of the total size of the dataset, graph databases excel at managing highly connected data and complex queries. With only a schema and a set of starting points, a graph database can explore adjacent data around these initial starting points, collect and summarize information from millions of nodes and relationships, and leave any data outside the search scope unchanged.

One of the few ways to build graph databases is a property graph model, where data is organized into nodes, relationships, and properties. Here we give a description:

Node A node is an entity. It can contain any number of properties. Nodes can be labeled with labels to represent different roles in the domain. Node labels can also be used to attach metadata, such as index or constraint information, to certain nodes.

Property Property is a key-value pair, and is stored on nodes or relationships. A node or a relationship can have many properties.

Relationship A relationship, also termed edge, defines a directed and semantically related connection between two entity nodes, such as *is the creator of* in the triple *(Rembrandt, is the creator of, the Night Watch)*. Relationships always have direction, type, start node, and end node. Like nodes, relationships can have properties. Sometimes, relationships have some quantitative properties, such as weight, cost, distance or rank.

We choose Neo4j[4], an open-source graph database management system, as our graph database. Neo4j has various plugins to make the implementation, such as graph import and graph algorithms, painlessly. Besides that, thanks to various development toolkit for neo4j, the graph data of Neo4j can be manipulated easily either through browser, API or Python.

2.2.3 Cultural Heritage Knowledge Graph

Collecting and building a graph knowledge base is one of the prerequisites for establishing our agent. With the development of information technology, many museums have digitized their collections to store them and present them to the audience. On the other hand, applications relevant to the museum get benefits from the digitized heritage data as well. To the best of my knowledge, there are few English knowledge graphs specifically constructed for cultural heritage that are available to the public, Here we list options that could be used in our project.

A few services of knowledge graph for art works are open to the public, for instance, Europeana[5]. Besides that, the open encyclopedic knowledge graph which contains art and heritage collections, such as Wikidata[6] and DBpedia[7], could also be the source of data.

Wikidata Wikidata consists of the common knowledge of Wikipedia entries by extracting structured data from that page. Wikidata is a large database of knowledge that is readable, writable and editable both for machines and humans. Different from DBpedia, Wikidata allows users to participate in the data editing and management. Another notable feature of Wikidata is that it is multi-language, that is, data input in any language will be immediately displayed in other languages. What's more, Wikidata attaches great importance to the source of data and marks it. Currently, it includes around 93 million entities (pages), and this number is still growing.

Wikidata is released under a Creative Commons CC0 License that allows data reuse while allowing users to copy, modify and distribute the data without asking permission, even for commercial purposes. Based on a free license and massive amounts of sourced data, Wikidata has become one of the most popular data sources on the Internet. Plenty of research has been carried out using it as a database, and many applications developed based on it.

In this project, we mainly focus on art-related data, or more specifically, painting entities within Wikidata. Currently, There are in total 539,416 paintings entities in Wikidata.

DBpedia Wikipedia's articles are mostly free text, but they also contain a variety of structured wiki-style information. This information includes Info box templates, category information, images, geographic coordinates, links to external interfaces, disambiguation interfaces, redirects between interfaces, and links to interfaces in different languages. The DBpedia project has extracted structured information from Wikipedia since 2007 and turned it into a rich knowledge base. The data is stored in RDF format and is queryable. Now, there are 6 million entities and 9.5 billion RDF triples. It is also free to copy, distribute transmit and adapt the data in DBpedia.

[4]https://neo4j.com/
[5]https://www.europeana.eu/en
[6]https://www.wikidata.org/wiki/Wikidata:Main_Page
[7]https://www.dbpedia.org/

Europeana Europeana aims to integrate representative cultural heritage resources in Europe by building a unified network platform, provide one-stop browsing and retrieval services for people to understand European history and culture, and achieve wider dissemination and sharing of European digital cultural resources. Since the project launched in 2008, now there are over 3,000 institutions and 50 million records of artifacts included in Europeana. It offers various APIs[8] for developers to build applications with their collections.

Europeana organizes and structures the data in the form of Europeana Data Model (EDM). To use its APIs, a basic understanding of EDM is necessary. What should be noticed is Europeana contains lots of non-English entities.

Our choice Among the above three options of data sources, we choose to query those linked open data we need on Wikidata, since Wikidata has more friendly interface to query data and the library of our graph database supports to extract and store data from Wikidata smoothly.

2.2.4 Query RDF data

SPARQL Language RDF is a data model, and to query the data in RDF format, SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) is used. The whole Wikidata RDF dataset is too large to store on the local server, meanwhile only the triples related to paintings of art in Wikidata are needed in our project, we need to use SPARQL to query triples desired through Wikidata query service. The service puts Wikidata and SPARQL together: you enter a SPARQL query, which runs against Wikidata's dataset and display the results. RDF Data Access Working Group of the World Wide Web Consortium firstly standardized SPARQL and recognized it as one of the key technologies of semantic web in 2008. From the full name of SPARQL, we can know that it consists of two parts: protocol and query language. For the query language, similar to using Structured Query Language (SQL) to query a relational database, SPARQL is used to query and manipulate data in RDF format. As for the protocol, it means that we can transmit queries and results between the client and the SPARQL endpoint through the HTTP protocol, in other words, the SPARQL endpoint can accept SPARQL queries from clients and return results.

Cypher Language As a graph query language, Cypher[9] is used to perform query and other operations on Neo4j, such as creating nodes, importing RDF data to Neo4j and finding the shortest path between entities. Inspired by SQL, Cypher has similar keywords and constructs as SQL, but it is designed and optimized for manipulating graph data.

In this thesis, we use SPARQL language to query RDF data from Wikidata query service. The Cypher language, as proposed by Neo4j, is used to manipulate the knowledge graph in the Neo4j graph database, such as search for an entity or path. Cypher is designed to deal with entities and relationships containing attributes in a knowledge graph, while SPARQL language can process data in RDF format.

2.3 Knowledge Based Conversational Agent and Question Answering

A Knowledge Based Question Answering (KBQA) component should be included as a part of our agent. As we mentioned in the first research question, we want to build a knowledge graph driven conversational virtual museum guide which is able to answer factoid questions about the museum from visitors. The task of a KBQA system is to extract the answer from RDF data based on the natural language question.

Demoing Platypus [3] is a multilingual question answering platform for Wikidata, which uses logical representations to represent questions and is able to combine several partial representations to answer complex questions. In this model, logical representations are generated by a grammatical analyzer and a template analyzer.

During the interaction between agent and user, it is a challenge to interpret the meaning of utterances correctly due to the ellipsis phenomena [4]. For instance, when a user first asks *"Who is the President of the United States"* and gets the answer, then a following question could be like *"Where*

[8] https://pro.europeana.eu/page/apis
[9] https://neo4j.com/developer/cypher/

was he born?". In the following question, the ellipsis of the entity *"he"* represents *"the President of the United States"*. To cope with it, Guo et al. [4] introduced a dialogue memory management component. When generating the logical forms of the utterance, the dialogue memory component can utilize the context entities, predicates and action subsequences.

To and Reformat [5] focus on building a template-based question answering system containing linguistic terms, such as bigger or smaller. The system contains a template generation mode and an answering question mode. For the first mode, dependency trees from SPACY are used to interpret the input question, and then the system obtains the answer through question template matching and SPARQL query template generation. As for the second mode, the system first instantiates the template and then maps lexical expressions into KG primitives to answer the questions with linguistic terms.

Plepi et al. [6] stressed the weakness of normal stepwise approaches for building a conversational question answering system: first, the errors of upstream subtasks such as entity linking, detection are propagated to downstream ones e.g., semantic parsing, and second, supervision signals can not be shared among the models when models for subtasks are learned independently. They introduced a CARTON (Context trAnsformeR sTacked pOinter Networks) model, which is a multitask semantic parser for conversational question answering over knowledge graphs. The framework handles semantic parsing using the context transformer model and the remaining tasks such as type prediction, predicate prediction, and entity detection are handled by stacked pointer networks. In short, first, a Transformer-based contextual encoder finds the representation of the current input context of the dialogue. Then, a logical decoder generates the pattern of the logical forms. Last, the stacked pointer network initializes the KG items to fetch the correct answer.

For the topic of knowledge based conversational question answering, Steinmetz et al. [7] recently proposed an approach to deal with slight shifts of context by analyzing follow-up questions for context shifts rather than keeping a certain node distance within the knowledge graph. And they also propose a new conversational QA evaluation data set based on DBpedia.

In our project, we focus more on the recommendation. Thus, as for the KGQA component, we aim to implement a template-based module to answer users' factoid questions.

2.4 Recommendations in Conversational Agent

Providing suggestions based on the user's interest is another goal for our project. With the development of LOD, thousands of digital cultural heritage collections are open to users. When visitors interact with our virtual conversational museum guide, we want to share more knowledge with them than just the basic information of those exhibits in the museum. However, because of the huge amount of information available in the database, the user's experience may be spoiled if they try to retrieve items by themselves. Recommendation technologies can suggest items that are likely to interest the user. Apart from that, user modeling, as a mean for personalization, can make the recommendation more accurate and relevant to the interest of the user.

2.4.1 User Modeling in a Conversational Agent

User modeling for museum guides [8] focuses on the challenge how to use personalization to select information from the great number of available exhibits to present to museum visitors. The authors pointed out that, in the museum environment, the basic premise of achieving personalization and adaptability is "nonintrusiveness". In order to keep the visitors focusing on the exhibit itself in the museum, any intrusive actions such as asking the visitor to fill a questionnaire about their preference should be avoided. The information needed for user modeling should be collected during the visitor's interaction with the exhibits in the museum. For instance, the questions asked and the total time spent at every location can be used for user modeling. What's more, these interaction records can be summarized as a report to present to the visitor.

Thompson et al. [9] presented Adaptive Place Advisor (ADA) to give personalized recommendations during conversation. They stated that user models in general applications, such as search engines or shopping sites, aim to improve accuracy or recall. But the goal of user modeling in conversational agents is to enhance the ability to track the goal of the user during the process of dialogue. User models are used to represent a group of users who have similar behaviors or individuals by learning from their information or past behaviors. To collect the information or personal preference, two approaches are given: directly through feedback and unobtrusively inferring. The former explicitly

collects user's preferences by asking the user to provide ratings for items or finish a survey about the user's preference, while the latter infers user's preference by examining the behaviors. ADA applied the latter method, which requires less effort for the user and lightens the user's burden compared with the direct feedback method. They designed a long-term and fine-grained probabilistic model to learn and represent the user's preference based on the responses and choices of the user in conversation. During the long-term interaction process, this model is updated every time the user reacts, to give more stable suggestions.

It is a common method to obtain the user's preferences by examining attributes or functionalities of items. Jannach [10] designed a conversational agent to assist the customer to buy a camera. The agent asks the user questions about each predefined attribute in the user model, and then gives recommendations based on the answers to suit the user's purpose. This method works well for task-oriented conversations, such as booking a hotel or shopping. In the context of a museum guide, it is not mandatory for a user to explore the knowledge base. As a result, asking too many questions about attributes of art items to give recommendations may annoy the user. So the questions should either be short or easy to answer for the user, for example, only one or two questions are asked, and the user just need to click yes or no to answer rather than reply with a whole sentence.

But sometimes the items do not have these pre-defined attributes, or it is laborious to collect the user's preference on all kinds of items. Rana and Derek [11] introduced a conversational recommender system which leverages unstructured features, such as keywords and tags from the user's feedback. When the user mentions a specific preference of items repeatedly, the model finds similar items as recommendation. On the one hand, this method does not bother the user, on the other hand, it may give less accurate predictions on user's preference.

Apart from forementioned models, some works focus on dealing with a cold-start situation. The problem is, when a user starts a new conversation, the user profile is totally empty and the conversational agent has no idea about the user's preferences. A common strategy is to recommend the most popular items as an attempt. Narducci et al. [12] addressed this problem by collecting the preferences of a user community. In our project, we do not want the user to skip the exhibitions, so, before giving a recommendation, the virtual conversational guide should always have some conversations with the user about paintings displayed in the virtual exhibition, and the user profile can be built at this time. Thus, the cold-start situation does not affect the user's experience much.

2.4.2 Embedding-based Recommendation on Knowledge Graph

User modeling methods we discussed above can be utilized to extract the preferences of the user, but another problem we meet for giving recommendations is how to select the appropriate contents for recommendation among all the available items. To address this problem, two methods are usually considered: collaborative filter and content-based recommendation [13]. Collaborative filter utilizes behaviors of past users to predict the interests of current users. For example, if a lot of visitors show their interests in the paintings from Vincent van Gogh and Rembrandt, then it can be inferred a new visitor who likes van Gogh is likely interested in Rembrandt and our system would suggest paintings from Rembrandt. As for the latter method, the content-based recommendation defines several features of the item and determines the user profile as a list of importance to these features by leveraging this user's past interaction with the system. For instance, if a visitor shows interests in the paintings from the Baroque, since the Baroque is a value of the features we defined, then the system would suggest more paintings from the Baroque period. For our project, the collaborative filter does not suit well due to the cold-start issue.

Our conversational agent is knowledge based, thus we focus on the content-based recommendation technologies on knowledge graph. Previous work on knowledge graph based recommendation can be mainly divided into three categories: embedding-based methods, path-based methods and unified methods [14]. Most of the embedding-based methods use the knowledge graph implicitly to learn the representation of items, while path-based methods utilize the connection between user and items in the graph. Given that there are no user entities in our knowledge graph, we choose to implement embedding-based recommendation methods.

Embedding KG components i.e., entities and relations into a continuous vector space is a new direction to simplify KG operations while preserving the inherent structure of KG. At the same time, it is a basic and efficient method for giving recommendations on KG. The embedding techniques are mainly categorized into two groups roughly [15]. In this part, we first talk about translational distance

models and then the semantic matching models.

Translational Distance Models The translational distance model [16] utilizes distance-based scoring. They usually translate through relations, and calculate the score using the distance between two entities in vector space. The basic idea is to predict the target entity or relation using the sum of the other vectors in a triple. In this way, it transforms the learning of the representation of entities and relations into an optimization problem that minimizes the distance between the target vector and the sum of other vectors in a triple.

Semantic Matching Models Semantic matching models, also called semantic information based models [17], use a similarity-based scoring function. They measure the credibility score of a triple by matching the latent semantics of the entity and the relations contained in the vector space representation. In our knowledge base, each entity has several properties, and the values of these properties can be used as semantics to measure the similarity score between two entities.

The early models usually considered individual facts and ignored their internal association, which was not sufficient to capture deeper semantics for better embedding. Neural-network-models recently are proposed to use additional information such as path, type of entity and relation and temporal information to get better embeddings. Wang et al. [17] give a comprehensive summary about these neural-network-models.

Language Models Based Approaches Besides the aforementioned models, another group of models utilizes language models to embed knowledge graph, such as RDF2vec [18], Wembedder [19] and KGvec2go [20].

RDF2vec [18] shares a similar idea of Word2vec [21] to train the embedding model. Word2vec takes a set of sentences as input and trains the neural network using one of two variants: continuous bag of words (CBOW), or skip gram. In CBOW methods, the model is trained to predict the words when context words are given, as for skip gram, it uses input words to predict the context words. This idea also works for RDF data, and there are two steps to train a RDF2vec model. First, random walks are used to create sentences of RDF triples in the graph, second, those sentences are used to feed in the Word2vec model. In the vector space of RDF2vec, similar entities are closer, which is similar to word2vec model. And RDF2vec has a pre-trained model[10] on Wikidata and DBpedia.

Several works offer pre-trained models based on RDF2vec. Wembedder [19] is a simplified version of RDF2vec and trained on a Wikidata dump using word2vec. The RDF data from Wikidata is processed, and each processed RDF triple can be treated as a sentence with three words: subject, predicate and object. This triple can be regarded as a simple walk from a subject to the object in Wikidata. Wembedder offers the service to find similar items in Wikidata via the REST API[11]. KGvec2go [20] shares a similar idea with RDF2vec as well, but it has pre-trained embedding models for four knowledge graphs: DBnary, DBpedia, WebIsALOD and WordNet. And it also offers a REST API[12] to retrieve the vectors.

Though Wembedder is pre-trained on the Wikidata and has a REST API to access easily, its training process does not utilize the whole data of the Wikidata, which causes that many painting entities exhibits in our virtual museum are out of the vocabulary.

Besides that, Chen et al. [22] propose Fast Random Projection (FastRP), a node embedding algorithm to project the graph into a lower dimensional space. The author states that it is 4,000 times faster than Node2vec [22], one of the classic node embedding methods, while has the similar or even better performance.

[10] http://data.dws.informatik.uni-mannheim.de/rdf2vec/models/
[11] https://wembedder.toolforge.org/most-similar/
[12] http://kgvec2go.org/

3 Related Work

In this section, a review of literature and related work is provided. These studies and works give a basics of knowledge of conversational agent. We first study two social chatbots, which have various skills to have conversations with human, their versatility shows the possibility for a versatile conversational virtual museum guide. Then, we conduct an unstructured review of conversational virtual museum guides.

3.1 Conversational Agents

A conversational agent is a dialogue system, which can not only understand what the user is saying, but also respond automatically in human language. The interaction can be conducted via text or speech. Here we give two examples, Alana and Xiaoice, two social chatbots which have multiple skills to conduct conversations with humans. They are good examples to mention here due to their comprehensive and advanced chatbot skills, and well-designed architecture.

Alana [23] is a socialbot in 2017 Alexa Prize challenge. This bot is an ensemble of different task and topic specific bots and combining rule-based and machine learning methodologies. The ensemble of bots consists of Persona bot, Newsbot, Factbot and Weatherbot etc. Alana is designed to have open-domain conversations with users and be informative and engaging via natural language and non-repetitive responses. The system first processes the input dialogue using multiple Natural Language Processing (NLP) tools and method, such as sentence completion, POS tagger, Named Entity Recognition (NER) and coherence resolution. Then, the preprocessed and annotated utterance and context is sent to all bots and the bots generate candidates for response. Candidates are processed and standardized to filter out repetition, profanity and single-word responses. Then, the final response is selected in three steps: bot priority list, contextual priority and ranking function. This system achieved great user rating scores and long dialogues averaging 10.7 turns.

In 2018, the enhanced Alana v2 [24] was introduced, with a deep Natural Language Understanding (NLU) pipeline and new features such as Named Entity Linking to find related information on Wikidata[13], clarification questions to disambiguate the entity and response retrieval from Reddit[14]. It is worth mentioning that the coherence bot of Alana can build a basic user model based on the user's preference shown during the conversation, so that it can determine to stay on topic or pick a topic that the user enjoyed in the past. As a social chatbot, Alana is versatile because of the ensemble of different bots, and it has a well-defined structure to process the user's utterances. Though Alana exploits the Wikidata as external related web information to enrich the response, it did not use the knowledge graph applications such as question answering and recommending knowledge to share with users.

Zhou et al. [25] introduced the development and structure of Microsoft Xiaoice in 2018, one of the most popular social chatbots. Xiaoice is designed to have three essential features: IQ, EQ and personality to support her chatting ability. IQ means Xiaoice masters many conversational skills, EQ shows she is aware of the mood and interests for the speaker and personality suggests Xiaoice has her own character, which is a sympathetic, kind, humorous young girl and her answer is in line with her personality. Besides that, an exclusive Dialogue Manager component is a part of the structure and it tracks the dialogue status and decides the next action based on the status updated. To achieve the general chat, Xiaoice first generates a response candidate set with the help of retrieving module and then ranks the candidates to determine the response. Though Xiaoice is capable to carry out chat with extensive skills and topics, her goal is to complete as many rounds of chitchat as possible with the speaker, rather than pass on knowledge through the conversation. At the same time, she does not have skills of knowledge based question answering and recommendation, nor does she have a dedicated knowledge base about cultural heritage.

3.2 Conversational Virtual Museum Guides

As an application of conversational agents, virtual museum guides have been studied in past 20 years and, from an unstructured literature review, we found 9 conversational museum agents [26, 27, 28,

[13] https://www.wikidata.org/
[14] https://www.reddit.com/

29, 30, 31, 32, 33, 34]. Among them, [26, 35, 28, 29] are embodied museum virtual agent, while [30, 31, 32, 33, 34] are text-based museum chatbot.

In 2002, Almeida and Shigeo [26] designed a human-like interactive virtual museum guide, which can guide visitors to virtual exhibitions and answer basic questions by detecting keywords upon user's requests. As one of the earliest virtual museum guides, it has a pre-defined museum setting and limited conversation scenario.

In 2005, Kopp et al. [27] introduced a virtual conversational agent Max which could communicate with visitors face-to-face, provide them with information about the exhibition and conduct natural small talk conversations. They define two aspects, dialogue act and conversational function, to represent a dialogue from agent, and interpret the text and generate utterances using rules.

Then, Ada and Grace [28] were created as virtual museum guides to use at the Museum of Science, Boston. They have photoreal animation appearance and interact with visitors in natural language. The input audio is recognized as text and then mapped to a set of possible responses using a statistical classifier in the Natural Language component. After that, a dialogue management module chooses one response to perform. For virtual museum guides, it is novel to have twin guides with a human-like appearance to interact with visitors. What's more, the guides have a large set of responses (about 400). Nonetheless, they lack the ability to be informative to share related properties about the collections based on the user's interests.

Tinker [29] is a virtual museum guide agent, designed to engage museum visitors by building human-like relationships with them. It utilizes multimedia, such as animation, motion, hand gesture and text to interact with visitors and social behaviors such as self-disclosure and empathy to establish social bonds with visitors. The paper concludes that having an agent which can build relationships satisfies visitors and increases their engagement during visits. Though Tinker demonstrates basic ability to establish social relationships with humans, it can not deal with factoid questions.

Art-bots [30] is a text-based conversational agent, and it is designed to be a personalized story-telling museum guide in a messenger chat surface. The art-bots can play a role as the owner of the museum and create an engaging visiting experience through actions like storytelling and games. However, the art-bot is highly customized, which makes it painful to implement for another museum.

CultureERICA [31] is an intelligent conversational agent, and it assists user to explore the exhibits within Europeana[15]. Though it is not a virtual museum guide, it still works in the CH field. CultureERICA is built based on Dialogflow, and it utilizes an improved algorithm to retrieve exhibits through the API of Europeana. The high quality graphic interface and easy interaction make it attractive to the user and leads to a high degree of user acceptance. However, CultureERICA focuses on improving the ranking algorithm for retrieving, it does not take the user modeling and recommendation into account.

The Culture Chatbot project [32] is a funded project by EU to build cultural chatbots for museums. The project investigated various chatbot technologies and came up with Culture Chatbot[36], a virtual tour guide for the Jewish Historical Museum in Amsterdam. Rasa[16] chatbot platform was chosen as the platform for constructing the chatbot. The chatbot was created using NLU and ML approaches, as well as linked to DGraph[17], a knowledge graph service, for query. It allows users to search items through text conversation. But, it does not support to recommend items to the user.

Another conversational bot for museum is developed by Cartier Foundation for Contemporary Art in France [33]. This chatbot is developed with Dialogflow, and it can answer pre-defined questions, such as questions about the event and calendar, and it can provide information about artworks, exhibitions or artists to the user so as to guide the conversation. Though it allows free text input from the user, its training is limited and questions beyond its understanding will be answered with a pre-defined template.

Máximo the Titanosaur [34] is a chatbot in The Field Museum of Chicago, which represents the largest dinosaur that ever existes and makes text chat with the visitor. The chatbot is developed with Dialogflow, and designed to be funny and delightful. A review [1] states that, as a museum conversational agent, it can answer almost any questions in a natural way, by having long-term and continuous user testing and dialogue redesign.

Briefly summarizing, the previous virtual museum guides always have an appealing appearance or

[15]https://www.europeana.eu/en
[16]https://rasa.com/
[17]https://dgraph.io/

stories as a part of the museum, and they focus on using multimedia to have chitchat with visitors and offering basic information about the collections, museum and even themselves. The virtual guides select an appropriate response from the database or generate one with rule-based methods, whereas they are not able to answer factoid questions about the collections beyond the questions set or to give personalized recommendations such as other similar art works the visitor may be interested in. Also, another

Our virtual conversational agent is aimed to be informative and able to answer the user's questions and give appropriate suggestions. To the best of our knowledge, these functionalities are not seen in previous works on virtual conversational museum guides.

3.3 Choice of the Systems Approach

In this section, we give a summary of the state-of-the-art for text-based museum conversational agents in Table 1. Currently, some museum chatbots focus only on a specific museum, such as [30, 33, 34]. These kinds of chatbot can offer sufficient information about the museum to the visitors, such as FAQ and stories behind the exhibits, but they lack knowledge of artworks outside the museum. Other chatbots [31, 36] are mainly used as conversational search engines, which offer an easy and human-like interaction for users to find artworks as if they are talking to real people. They have access to the knowledge base, which supports them to be able to find artworks required by the user. However, the existing museum conversational agents we surveyed generally do not use knowledge graphs, or simply search in the knowledge graphs without fully exploring the possibilities of knowledge graphs in assisting museum conversational agents to share knowledge, for example, by Q&A and recommendation as we introduced in Section 2.

CA	Platform	Character	Engagement	Conversational Skills	Advanced Methods	Q&A	KG	Recommenation
Art-bots [30]	Facebook	Storyteller	Play games	Input keyword to bring the story forward	No	No	No	No
CultureERICA [31]	Dialogflow	No	Guided search, rich multimedia content	Input free-text or keyword to search and filter result	ML and NLU	No	Use Europeana API for query	No
Culture Chatbot [36]	RASA	No	Free-text search, rich multimedia content	Input natural language or press button to search	Avaliable but not used	No	Yes, DGraph for query	No
Cartier Foundation for Contemporary Art Chatbot [33]	Dialogflow	No	Rich multimedia content	Limited in free-text input, offer button and multimedia content to control the chat	Avaliable but not used	Only FAQ	No	No
Maximo the Titanosaur [34]	Dialogflow	Friendly, a dinosaur	Text only, response smart and natural	Have very well natural language chat	ML and NLU Well trained	Yes, only about the exhibition.	No	No

Table 1: A summary of the state of the art in text-based conversational museum agent

4 Conversational Agent for Virtual Museum

In this chapter, we talk about our design for building a knowledge graph driven conversational virtual museum guide. According to the research questions and previous discussion of background knowledge and related work of CA, we present the main requirements of the CA in this thesis and an overview of its work flow in Section 4.1. Section 4.2 shows the design and construction of the CH knowledge graph database, which drives the features of our CA. Section 4.3 offers basic concepts of Dialogflow and presents the dialog flow design of main features of our CA. Finally, Section 4.4 demonstrates the design of webhook services, i.e. the knowledge graph driven features and the user model, for the CA.

4.1 Requirements and Work Flow

Currently, the state of the art for text-based museum conversational agent is stated in section 3.2 Besides that, we decided to develop the CA with Dialogflow. As Table 1 shows, Dialogflow is used by some state-of-the-art chatbots, since it is easy to use and offers advanced features such as ML, NLU and sentiment recognition. And similar to Cartier Foundation for Contemporary Art Chatbot [33], we want to design our CA to be more proactive. One of the reasons is that, considering the huge amount of entities, relationships and attributes in knowledge graph, guiding users to ask questions and look for recommendation make it easy to start exploring the knowledge graph. Another reason is that our CA is based on a virtual museum, thus the CA could alert the users proactively when they are distracted, so that to make the user focus on the exhibition in the virtual museum. As a result, during the design of the dialog flow, we intend to offer suggestion chips (a format of rich response in Dialogflow integration, see an example in Figure 5) as options or examples to the user to help them start the conversation and let them follow the path by the agent to ask questions and ask for recommendation.

With the idea of developing knowledge graph based Q&A and recommendation for our CA, next we need to determine the approaches. First, for the Q&A, ML based methods need a large and valid corpora for the training process. Due to lack of CH Q&A corpora and the small number of entity types and properties types we have in the knowledge graph, we decide to use a template-based method. And as for the recommendation, we choose the content-based method, since the agent in our thesis is cold-start. FastRP is used because of its ease of implementation and efficiency.

Based on the discussion above and our research question, a summary of all the requirements for our CA is listed in the Table 2.

Goal	Requirements	Description
The CA should have knowledge about paintings and items related to it	Knowledge Graph	KG is the data source where CA could learn about the characteristic of paintings, and it supports the QA and Recommendation component.
The CA should have natural conversation with users	Dialogflow	Dialogflow is an end-to-end development suite for building CA, it is powered by natural language processing techs such as Named Entity Recognition (NER) for intent detection, and supports custom rich response, webhook and integration.
The CA is able to answer factoid questions from users	Knowledge Graph	The knowledge graph is used to perform the query.
	Question Answering	The question answering component can translate user's natural language questions to query language used to retrieve in the knowledge graph, and then formulate the result in natural language as a response.
The CA can offer personalized recommendation to the user without being intrusive	Knowledge Graph	The knowledge graph is utilized to generate graph embedding for recommendation.
	User Model	The user model is used to capture the user's interest during the conversation without asking their preference explicitly.
	Recommendation	The recommendation component calculates and rank the similarity scores, combines the scores with user model to determine the recommended entity.

Table 2: A summary of the requirements for a knowledge driven conversational virtual museum guide

Design of the work flow Given the requirements above, an overview of the workflow for our virtual conversational agent is shown in Figure 3. In our project, the conversations should follow the steps:

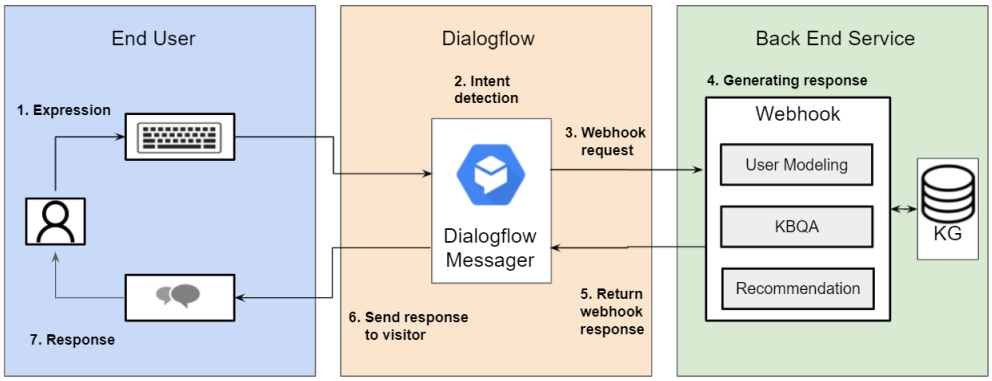


Figure 3: Design of the work flow

1. **User expression** The input utterances from the user should be text and the components in the webhook can only process text in our project.

2. **Intent detection** Here Dialogflow classifies the input utterances by the result of intent detection. The categories of intent need to be designed in advance in Dialogflow, which is introduced in Section 4.3.3. Dialogflow offers both rule-based and machine learning methods for the intent detection, and we use both of them for the intent detection. For example, for intent **Recommendation**, we added a training phrase "Show me similar exhibits" as a rule. Dialogflow's built-in machine learning expands the training phrase list with similar phrases. Other similar input, such as "Show me something similar" can be detected as a request of recommendation even though we never manually add this phrase to the list of training phrases.

3. **Webhook request** If the intent is detected and it requires accessing the webhook service for response, Dialogflow sends a request to the webhook service. Some types of intents do not need this step, for instance, the greeting.

4. **Generating response** Some simple responses can be generated by the pre-defined templates in Dialogflow, such as greeting the visitor. Besides that, other intents need to send request to webhook service to get response, for example, when the user asks a question or needs a recommendation. There are several components in the webhook service, and each of them fulfills its own purpose. User Modeling stores and updates the user's profile from the conversations and interactions when visiting. KBQA can answer factoid questions about the collection from the visitor. And the Recommendation module could suggest some items interesting to know for the user based on the user modeling result. The detailed description is given in Section 4.4.

5. **Return webhook response** Here the webhook service generates the response using templates. Since the dialogue mainly takes place in the context of a museum visit, the response patterns are limited, and we choose to use a template-based method to handle the response generation.

6. **Send response to visitor** Dialogflow sends the webhook response to the user. The rich response is supported, which contains image, list, options or external link for user.

7. **Response** The visitor receives the response and types their next input or end the conversation.

4.2 Knowledge Graph Database

This part describes how we design and construct the art knowledge graph used for our CA. In our project, we extract paintings and other entities related to the paintings from the Wikidata. Three parts are introduced respectively, i.e. the structure of graph data, querying using SPARQL on Wikidata and importing the result from Wikidata to our knowledge graph database.

4.2.1 Graph Structure

Before querying and collecting the data, we should first define the structure or schema of the graph data we will use in the project. After investigating the painting entities and other entities connected to paintings on Wikidata, as Figure 4 shows, we select 10 types of entity and 9 types of relationships to build the knowledge graph dataset. These entity and relationship types are selected as they are common properties of painting entities in Wikidata. A detailed schema of each type of entity and relationship can be found in Table 3 and 4. With the graph structure described above, we can store the RDF data queried from Wikidata in our local graph database. Since there are unstructured textual information and background stories for the 16 paintings provided by the Rembrandt House Museum, we should organize these data using the structure above and import them into our graph database.

Among the properties of entities in our knowledge graph, it should be noted that the property *sitelink* represents the number of how many times this entity is referenced by other entities in Wikidata. And *URI*, i.e. Uniform Resource Identifier, is a unique sequence of characters, which is used to identify an entity in Wikidata.

According to the function they serve, URIs could be categorized as Uniform Resource Locator (URL) or Uniform Resource Name (URN). URLs contain the location of the entities on the network. In Wikidata, an entity uses URL as its identifier and the schema of its URL looks like, for example, *http://www.wikidata.org/wiki/Q183*. Here, *Q183* is a unique identifier of Wikidata-object *Germany*, with it, we can locate the exact entity. As for URNs, a unique name is provided as the identifier instead of giving a link to locate the resource of the information.

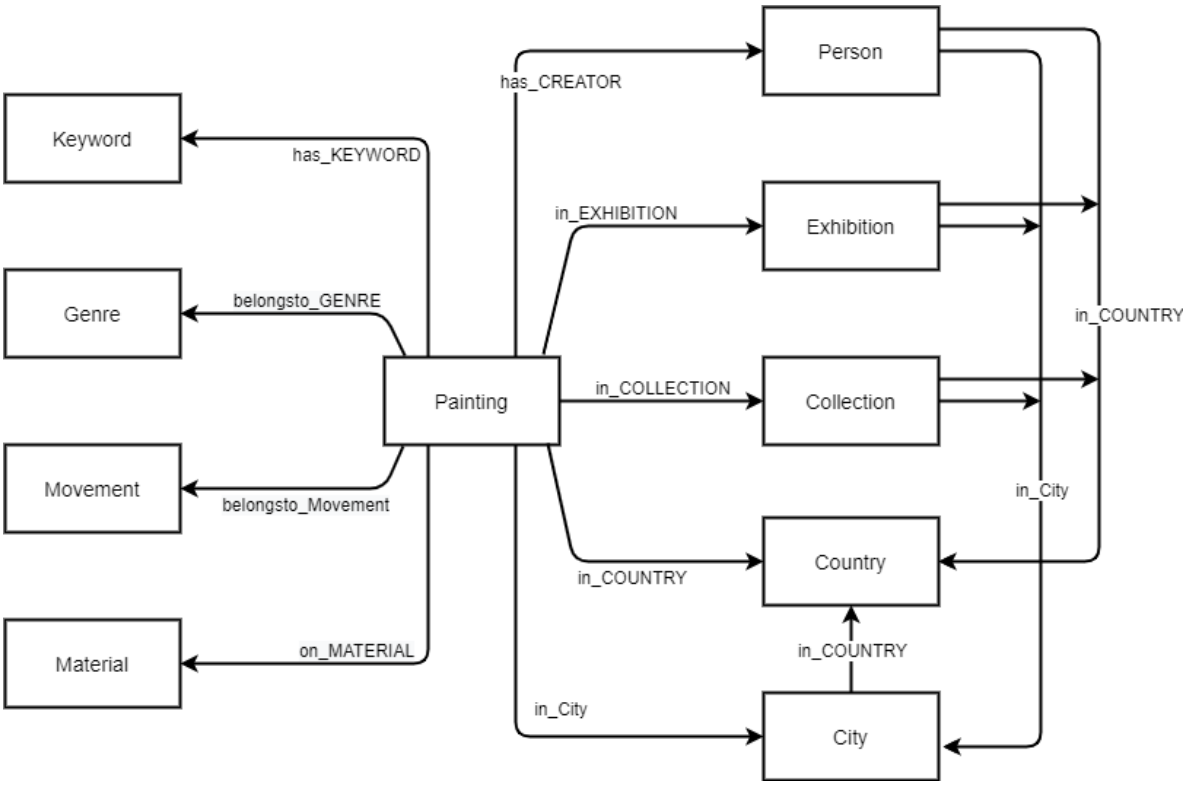


Figure 4: The structure of the graph.

4.2.2 Query on Wikidata using SPARQL

Before importing the RDF data from Wikidata as our knowledge graph database, running the SPARQL query on Wikidata Query Service[18] is advised, to prevent errors when importing. To understand how

[18]https://query.wikidata.org/

Entity type	Property	Count	Entity type	Property	Count
Painting	Name Description Image Date Height Width Sitelink URI	2990	Person	Name Description Image Sex Date_of_birth Date_of_death Sitelink URI	25
Collection	Name Description Image Date Website Collection_size Visitor_per_year Sitelink URI	334	Exhibition	Name Description Date Website Sitelink URI	127
Genre	Name Description Sitelink URI	29	Movement	Name Description Start_time End_time Sitelink URI	29
Material	Name Description Sitelink URI	46	Keyword	Name Description Sitelink URI	1919
Country	Name Description Sitelink URI	39	City	Name Description Sitelink URI	251

Table 3: A summary of entities and their properties.

the query works, for example, we have a triple *"The Night Watch, creator, Rembrandt"*, which means the Night Watch's creator is Rembrandt. In Wikidata, this triple consists of the subject *The Night Watch* (Q219831), the predicate *Creator* (P170), and an object *Rembrandt* (Q5598). This triple can also be represented by their URIs:

$$< http://www.wikidata.org/entity/Q219831 >$$
$$< http://www.wikidata.org/prop/direct/P170 >$$
$$< http://www.wikidata.org/entity/Q5598 > .$$

Prefixes When we query in Wikidata SPARQL service, these long URIs can be written in a more concise way:

$$wd:Q219831 \quad wdt:P170 \quad wd:Q5598.$$

In the above statement, *wd:* and *wdt:* are prefixes, which are the abbreviations of full URIs to make URIs short. *wd:* represents the entity in Wikidata, and *wdt:* is the property connecting the subject and object. To query the wanted entity, we can set the unknown value in the triple as a string beginning with a question mark, for example, if we want to query the creator of "The Night Watch":

$$wd:Q219831 \quad wdt:P170 \quad ?creator.$$

This query indicates to select the object that have a predicate of *creator* with a subject of *The Night Watch*.

Relation Type	Head Entity	Target Entity	Count
has_CREATOR	Painting	Person	6010
belongsto_GENRE	Painting	Genre	5474
belongsto_MOVEMENT	Painting	Movement	2106
in_COLLECTION	Painting	Collection	4944
in_EXHIBITION	Painting	Exhibition	632
on_MATERIAL	Painting	Material	5064
has_KEYWORD	Painting	Keyword	7742
in_CITY	Painting Collection Exhibition Person	City	
in_COUNTRY	Painting Collection Exhibition Person	Country	412

Table 4: A summary of relationships.

Label service Besides prefixes mentioned above, getting the name and description of the entity in English is also important when querying on Wikidata SPARQL service, since Wikidata is multilingual and our project is in English. The label service in Wikidata makes it effortless to get the name and description of the entity and filter the language at the same time, simply by adding the following code to SPARQL query:

$$SERVICE \quad wikibase:label \quad \{bd:serviceParam \quad wikibase:language"en".\}$$

Then for a variable, for example *?creator*, its label and description variables are named as *?creatorLabel* and *?creatorDescription* respectively. And Wikidata SPARQL Service will return the label and description values if we put these two variables in **SELECT**.

4.2.3 Import Query Result into Graph Database

With the properly working Wikidata query, we can start importing the results of the query into our local Neo4J graph database. Though Neo4j itself can not read the results from an online SPARQL retrieval service, neosemantics[19], a plugin of Neo4j, allows us to import and store RDF files from online service to Neo4j. Then we construct the graph as Figure 4, a basic summary of entities and relationships as well as their properties are listed in Table 3 and Table 4. The dataset we stored can be found in our project repository[20].

4.3 Design of Conversational Agent

Our project uses Google Dialogflow CX (hereinafter called "Dialogflow") as the framework for the conversational agent. In Section 2.1, we talked about what is Dialogflow and compared it with RASA briefly. Here, we will give a thorough description of Dialogflow and how we designed and implemented the chat flow.

4.3.1 Basic concept of Dialogflow

Dialogflow helps developers build a state-machine conversational agent, which understands user input with pre-defined "intent", and gives a pre-defined response or takes further action, such as utilizing webhooks to give a customized response, when matching a certain intention. Dialogflow offers a visualized console, thus the designing progress gets more intuitive and the dialogue status and flow can be tracked and controlled easily.

[19]https://neo4j.com/labs/neosemantics/4.3/
[20]https://github.com/LDLucien/dfcx_virtual_museum_guide

Before the actual implementation, we will introduce some basic concepts used by the conversational agent in Dialogflow. Table 5 shows a summary of the concepts used in this project.

Concept	Definition	Example
Entity Type	An entity type is a set for a class of things.	Painting is an entity type, it includes the names of all paintings in our knowledge graph.
Intent	Intent is used to categorize end user intent in a round of conversation.	Greeting, as an instance of intent, is triggered when user says hello or hi.
Page	Page is the state of the agent, which may contain multiple intents and fulfillment.	One page is for recommendations. On this page, the intention to terminate recommendation is triggered when the user enters an utterance similar to "Please stop recommending", and the agent replies to the user "No problem, you can visit the rest of the exhibition!", and then deactivates current page and the next page becomes active.
Webhook	A webhook hosts a backend service.	When user's input is recognized as a factoid question, the question will be passed to the webhook service to classify the question, query for result and fill the answer in a reply template as response.
Fulfillment	Fulfillment is the agent's response or action to user's input.	The fulfillment when user asks for a tutorial about the agent is replying with a static tutorial list. When a factoid is asked, the fulfillment uses a webhook service to generate a dynamic response.

Table 5: A summary and example of concepts in Dialogflow used in our conversational virtual museum guide.

Entity type An entity type is a set for a class of things. Dialogflow extracts keywords from the user input by entity types. Some common types of entity are pre-defined in Dialogflow, such as times, colors, called system entities. Besides that, we can define custom entities for matching user input.

In our project, we create entity types as what is listed in Table 3. Then, for each entity type. we extract the names of the entities and upload them to Dialogflow through Google Cloud Dialogflowcx v3 API.

Intent Intent is used to categorize end user intent in a round of conversation, when an end user types or speaks something (called "end user input"). Table 6 lists all the intents we defined and used for our CA in Dialogflow. When there is an input, Dialogflow compares the input to the intents in the list to find the best match. Intent training phrases are example phrases that could be end user input. If the end user's input has the highest similarity confidence to one training phrase of an intent, Dialogflow matches the input with that intent. Luckily, we do not need to define all possible examples, because Dialogflow's built-in machine learning capabilities extend the phrase list with other similar phrases. The training phrases may contain entity types we defined, and these entity types can be annotated automatically by Dialogflow or modified manually by us.

If we want to train an intent to detect utterances which include painting entities, for example, "The Night Watch" is an instance of the entity type **Painting** and we can add "Who created The Night Watch" as a training phrase, then the entity in the training phrase can be annotated by Dialogflow or manually. After training, all other input utterances that contain painting entities and are similar to training phrases can be detected by Dialogflow.

Page Dialogflow uses the state machine approach to design a conversational agent, and each state is represented through a page. Each agent can define many pages, but only the pages currently in use are active at any given moment in the conversation. When the page becomes active, the agent will execute the actions defined in the page in a certain sequence. And the actions within a page could be sending pre-defined replies, waiting for user input, calling webhooks, and so on.

For example, at a page, we could define several intent routes. Intent route is a kind of routing, which is invoked when the end user input matches an intent given in this page. Dialogflow trains models for each intent and when there is an input utterance, it calculates each intention model's confidence score with respect to the input and the intent with the highest score is matched. It deals with the

Intent Name	Training Phrase
Exhibit.display	I would like to know about *King Caspar*. Tell me about *Two moors*.
Question.have	I want to ask questions. How should I ask questions?
Question.answer	Who is the creator of *The Night Watch*? What key elements are in *The Night Watch*?
Recommendation	Show me similar exhibits like this one. I want to know some similar exhibits.
Recommendation.attribute	Another one related to *oil paint*. Tell me another entity related to *Rembrandt House Museum*.
Recommendation.nomore	Stop the recommendation, please. That's enough, thanks.
Recommendation.new	Do you have another one? Could you show me one more suggestion?
Recommendation.connection	Why this one? What's the connection between them?
Visit.finish	Finish the visiting. I want to stop the visiting now.
Universal.yes	Yes. Why not. Sure.
Universal.no	No, thank you. I am good. It's enough.

Table 6: The intents list and samples of their training phrases. The italic name in training phrases indicates that it is an instance of an entity type, for example, *King Caspar* is an instance of **Painting** and *oil paint* is an instance of **Material**.

current intents from the end user and generates responses. After the route is called, an optional page transition is used to change the current page to the next page assigned by us.

Webhook A webhook is a service that hosts our backend components. During a dialogue session, with the help of webhooks, we can use data, such as entity and intents, extracted through Dialogflow's natural language processing to generate dynamic responses at the back end.

To make our backend components available as webhook on Dialogflow, they must have a publicly accessible URL for requests. At present, these components are hosted on our local machine in Flask[21] framework. And we use Ngrok[22], which is a free application that exposes local server ports to the Internet, to build public URLs for our backend components.

Fulfillment In Dialogflow, during each round of the conversation, the agent must respond to the user by answering questions, asking for information, or terminating the conversation. What's more, if a webhook is activated, the agent may also need to send a request to the webhook to generate dynamic responses. Fulfillment is used to organize and accomplish all of these operations. Fulfillment may contain a static text response or a dynamic response generated from a webhook. Also, in a round of the conversation, it is possible to call multiple fulfillments as response. Fulfillment can be called when a new page is active, or when an intent route is called.

4.3.2 Integration

Integrations in Dialogflow provide the interface to the end user. We use Dialogflow Messenger, a built-in integration of Dialogflow. The Dialogflow Messenger (hereinafter Messenger) integration provides a customizable chat box that can be embedded into the website. After embedding the Messenger code, the chat box appears in the bottom right of the web page and the end user can open or close it, as Figure 5 shows. The appearance of the chat box can be customized, such as color, name and icon.

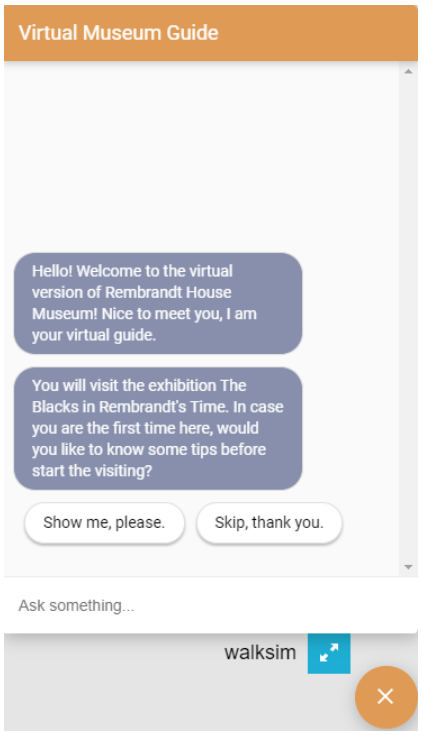


Figure 5: Embedded Dialogflow Messenger at webpage.

In Messenger, when we create fulfillment, it could be text responses and custom payloads. Text responses are used for text-based responses, and custom payloads are used for rich responses. The

[21] https://flask.palletsprojects.com/en/2.0.x/
[22] https://ngrok.com/

custom payloads enrich the conversation by providing a variety of responses templates, including card, image, button, list, suggestion chips and so on. What is worthy to mention here is the suggestion chips, which offer the opportunity for the end user to simply choose an option and then get a response. When a user chooses an option provided by the agent as input, the agent should be able to detect the intent more accurately, as the given options in suggestion chips are template based and similar to the training phrases of the agent's intents.

4.3.3 Design of Dialog Flow

In our project, the virtual museum conversational agent is designed and built for a scenario where a user visits the online virtual museum. Thus, different from an open domain chatbot or conversational agent who is able to carry out all kinds of dialogues, our agent focuses on the scene of museum visiting. Given this condition, here we give a description about the design of the dialog flow in detail, including the main function, the intents, and the fulfillment from the CA.

Our agent has six main capabilities, greeting and tutorial, exhibits displaying, question answering, recommendation, report generating and coping strategy towards problem. We give descriptions and figures below of how each of them works, respectively.

Greet and Tutorial Before the end user starts visiting the virtual museum, our agent should give an instruction about basic operations and tips during the visiting. Figure 6 gives the dialog flow about this function. As the user opens the chat interface, our agent first greets the visitor as well as introduces itself. At this point, our agent also asks if the user needs a tutorial about the virtual museum and conversational agent. If the answer is positive, the agent shows a short list of tips to the user. Otherwise, the user can skip this tutorial and enter the waiting stage.

The **waiting stage** means the status where the agent waits for the input from the user to activate any one of the five flows below.

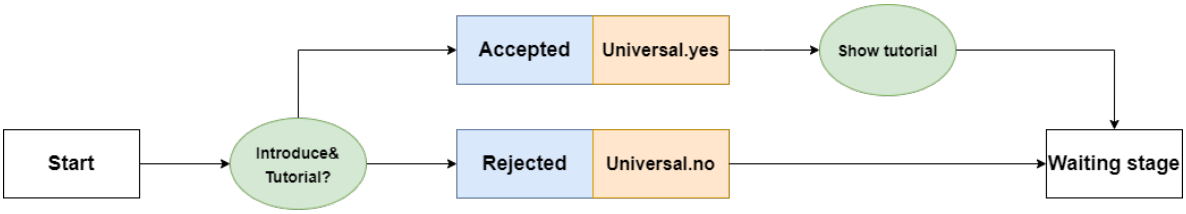
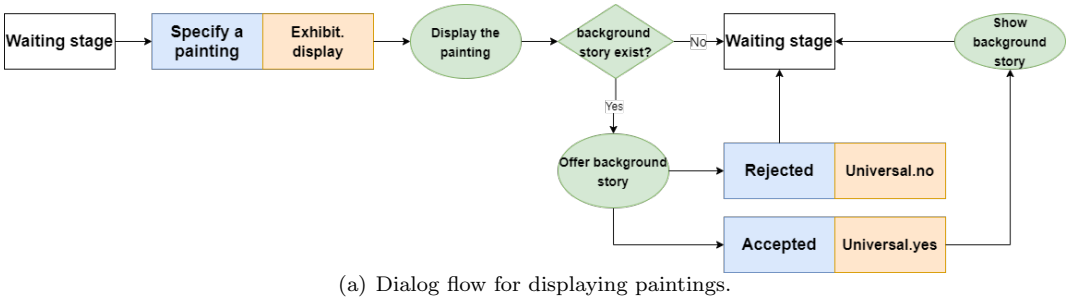


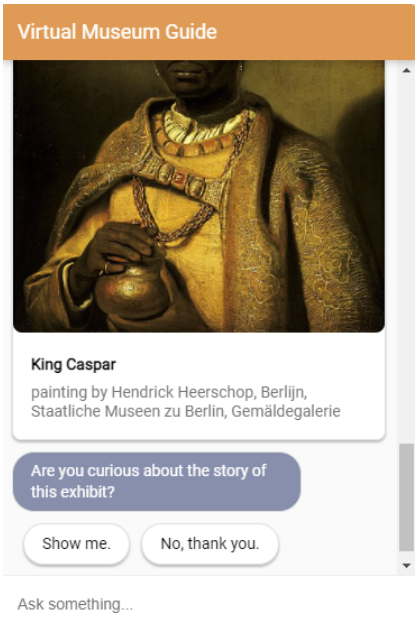
Figure 6: The dialog flow for introduction and offering tutorial. The blue box means the user's action, orange box represents the intents and green ellipse shows the fulfillment action, i.e. response from the agent.

Display Exhibits After the greeting and tutorial above, the user can start the visiting in the virtual museum. At the same time, our agent is in the waiting stage, awaiting the input from the user. To start the exhibit displaying function, the input should be an utterance related to an exhibit or a painting which exists in our knowledge graph database, otherwise the agent will not understand it. The dialog flow of this function is presented in Figure 7(a). To train the intent *Exhibit.display*, we create training phrases, as Table 6 shows. When the intent *Exhibit.display* is matched, the agent will send a request to webhook service, this request contains the detected intent with a confidence score and entity extracted in the utterance. According to the information in the request, our webhook service queries the desired entity in our graph database and organizes the query result in the form of a rich response of the Messenger, then sends the response back to our agent. The agent shows the response to the user afterwards. The response here consists of the three properties of the entity in the art knowledge graph database, image, name and a short description. In addition, by clicking on the response, the user is directed to the entity's Wikidata page. Figure 7(b) displays an example response.

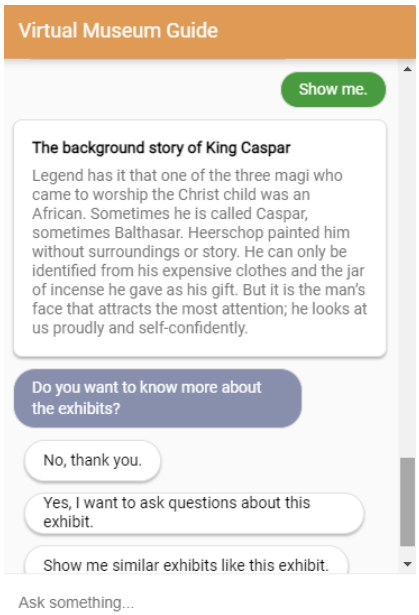
Specially, if the user asks about an exhibit within the virtual museum, such as *Two moors* in figure 7(b), our agent will ask if the user wants to hear a background story about the exhibit. The user can choose to accept and see the story or decline. When the user agrees to see the story, the agent will send a request to webhook and then return the story as a response, for instance, the response in Figure 7(c).



(a) Dialog flow for displaying paintings.



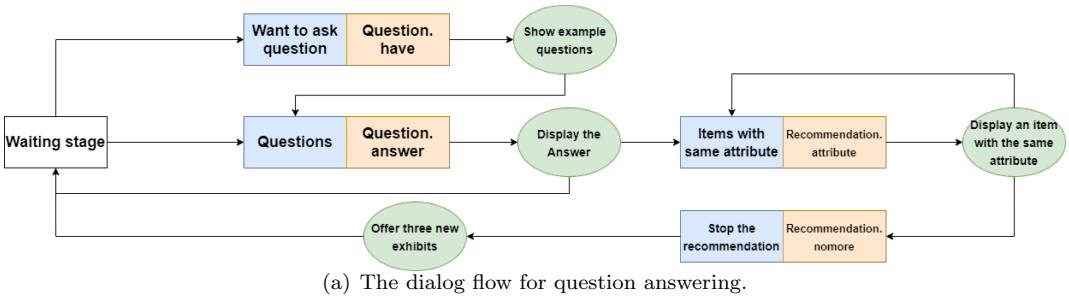
(b) Response when input "Show me King Caspar".



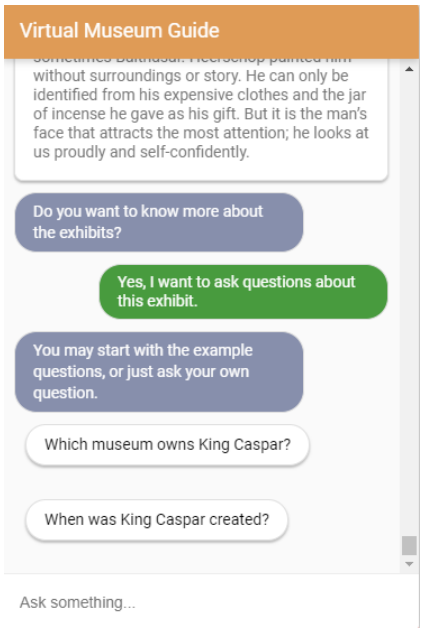
(c) The agent shows background story of the King Caspar.

Figure 7: Dialog flow for displaying the exhibits and examples.

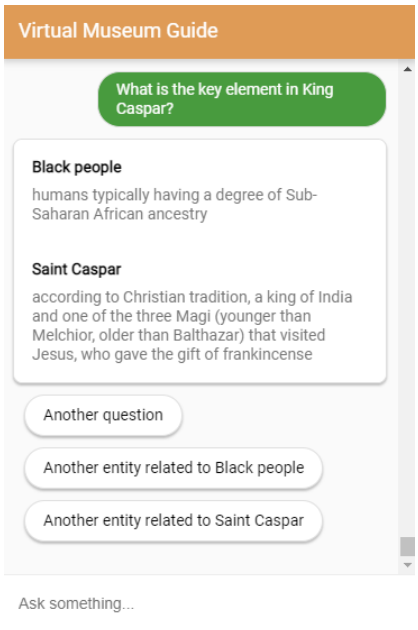
Question Answering With a knowledge graph database, we could answer the questions from the end user. The description and training phrases for intent *Question.answer* are listed in Table 6. When the intent is detected, our agent sends the request containing the question and detected entity to the KBQA component of the webhook service. The KBQA component utilizes a template-based method to categorize the question and then formulate the Cypher query to fetch the answer in the knowledge graph database. Finally, a response contains the answer is generated and returned to the agent. A detailed design of the KBQA component can be found in Section 4.4.1.



(a) The dialog flow for question answering.



(b) CA shows example questions to the user.



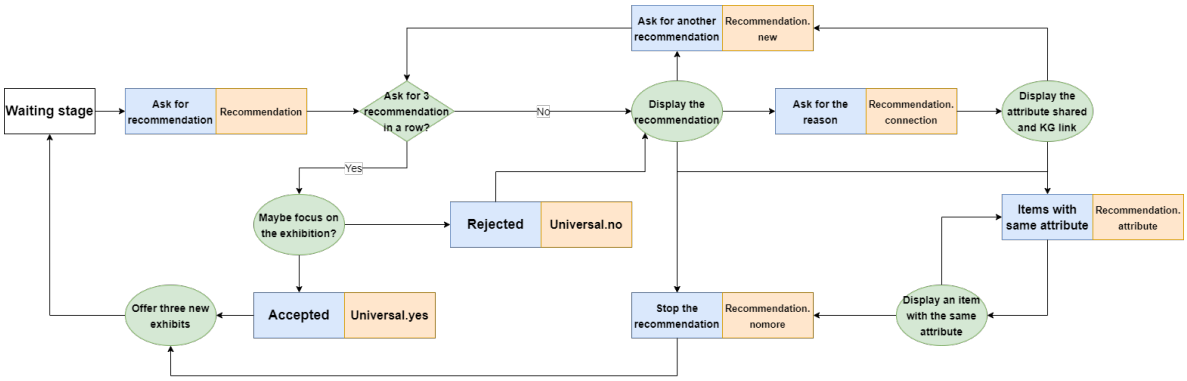
(c) CA offers options for other paintings with the same attribute.

Figure 8: Dialog flow for question answering (a) and examples of offering options for example questions and related entities (b, c).

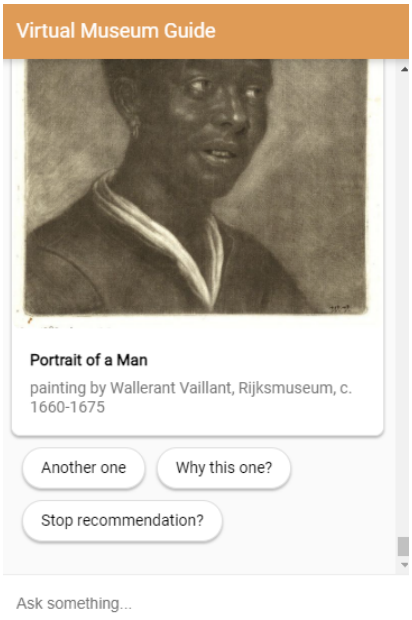
Besides asking the questions directly to the agent, if the users does not know how to start asking a question to our CA, they can tell the agent that they want to ask questions, and the agent can offer them example questions to start, as Figure 8(b) shows. After the CA has answered the question, it can offer options for the users to select if they want to ask a new question or would like to know other paintings with the same attribute. For example, in Figure 8(c), when we ask the CA "What is the key element in King Caspar", after answering the question, the agent offers us three options to choose.

Recommendation One of the huge benefits of using a knowledge graph is that, within a graph, we could explore data through the edges from one starting point. Similar to a redirected link on Wikipedia, we can find entities related to the current entity but in an intuitive and structured way. However, exploring in a huge knowledge graph may also be aimless and troublesome, when there are tons of possible paths to walk through. Thus, we want to enable the agent to decide and recommend entities to the user to help them explore the knowledge graph during the visiting without being lost.

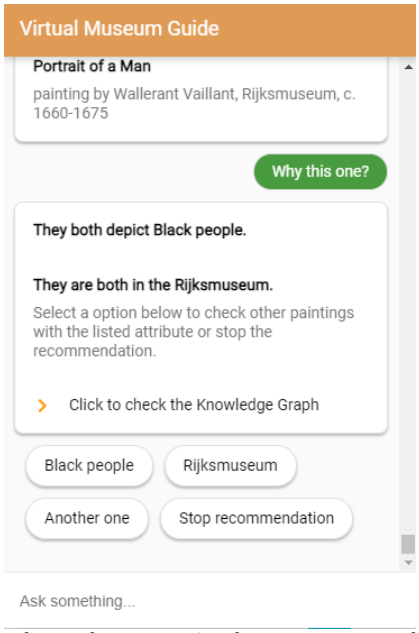
The intent *Recommendation*, shown in Table 6, can trigger the actions of recommendation. Our agent sends a request to the webhook service, the recommendation component finds a suggested entity based on the similarity scores of graph embedding and user profile. A detailed design of the recommendation service for our agent is presented in Section 4.4.3. Then, the webhook service generates the response and sends it back to the user.



(a) The dialog flow for recommendation.



(b) CA offers options after giving a recommendation



(c) CA shows the connection between suggested entity and starting entity as explanation.

Figure 9: Dialog flow for recommendation (a) and examples of options after recommendation and show the connection between entities as explanation (b, c).

Along with the recommendation, the CA can offer three options, as Figure 9(b) shows. The user can select to get another recommendation, ask for the reason of the recommendation, or stop the recommendation.

If the user asks for the reason, the agent will send back the shared attributes between the recommendation and the starting entity, an example is given in Figure 9(c). Besides that, a visualization of the knowledge graph, containing the starting entity, recommended entity and relationships between them, is generated and can be opened by clicking the link. Figure 10 gives an example of the visualization.

In addition to what we describe above, to make the visitor focus on the virtual museum rather than spend too much time viewing recommendations, an automatic alert is designed in this flow. When the visitor asks for recommendation three times without a break, the CA will remind the visitor whether to stop recommending and follow with the virtual exhibition. The user can select to stop recommendation or keep viewing the next suggested entity.

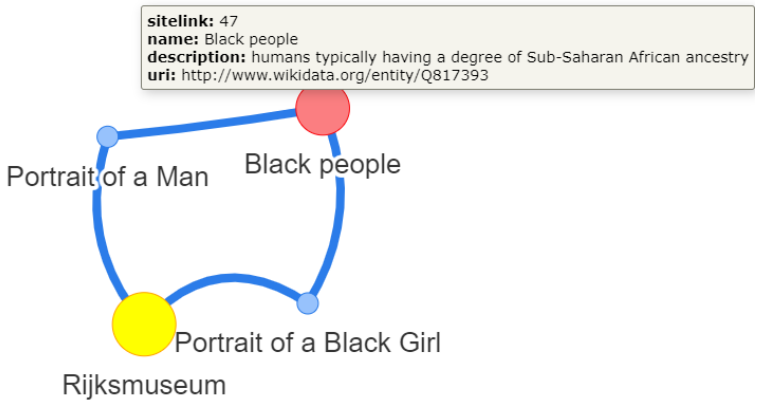


Figure 10: The visualization of knowledge graph given in Figure 9(c). The circle represent an entity, and the size of the circle is defined by the sitelink. The color of the circle means entity type.

Report generating At the end of the visit, we provide the user with a report summarizing the user's interactions and chats history with the museum guide. When the CA detects the intent **Visit.finish** as shown in Figure 11(a), it calls webhook service to generate a report based on the template. The report, for example in Figure 11(b), expresses the number of exhibits and paintings appeared in the conversation history, as well as the amount of types of painting's attribute the user asked.

Error Handling Strategy When communicating with a conversation agent, problems or failures are inevitable because the user's input is so diverse. In our project, such problems may be less frequent because our usage scenarios are limited to museum visits, and in many cases we provide visitors with options to help them engage in conversation, such as providing sample questions when users want to ask questions.

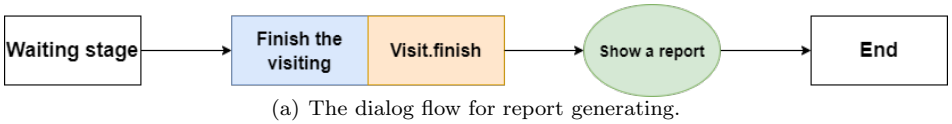
However, when there is no guarantee that users will follow the designed flow of conversations, it is essential for CAs to have coping strategies with problems. We present two examples strategies of our CA in Figure 12. In Figure 12(a), when a user's input can not be understood by the CA or is beyond the capabilities of it, the CA prompts the user that it has no way to do it, and then presents the user with a list of his capabilities. In another example, in Figure 12(b), when the CA executes the query for the user but has no results, it alerts the user that the relevant entity is not found in our knowledge graph database.

4.4 Design of Webhook Services

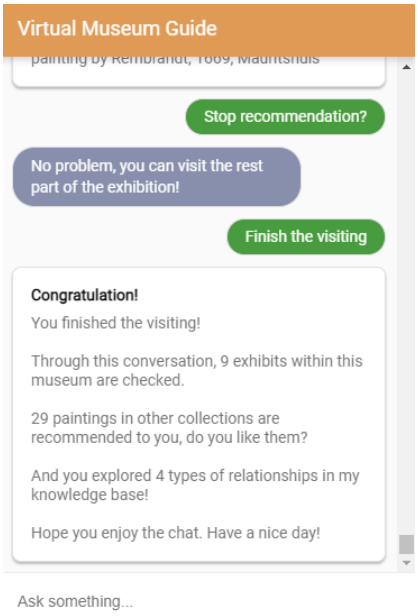
In this part, we mainly discuss the design of webhook services for our conversational virtual museum guide, consisting of a template-based KBQA, recommendation and user modeling.

4.4.1 Template-based Knowledge-based Question Answering

As we discussed in Section 2.3, a KBQA system is to extract the answer from RDF data based on the natural language question. In our project, a template based KBQA system is designed and used as a webhook service for our agent, which enables the agent to answer factoid questions about exhibits and painting entities stored in our knowledge graph database. In Section 2.3, we talked about template based and Machine Learning based KBQA systems, and for our project, we choose to design the KBQA system as template based. Considering the 9 different entity types in our knowledge graph database, the template based method can cover most types of questions about the entities and attributes they

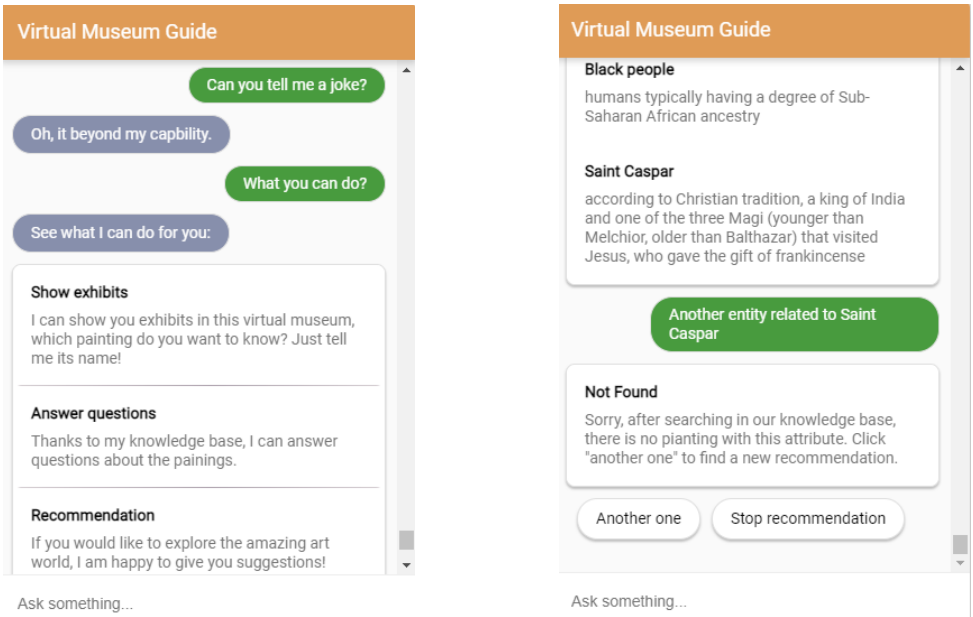


(a) The dialog flow for report generating.



(b) CA generates a report summarizing the user's chat history.

Figure 11: Dialog flow for report generating (a) and an example (b).



(a) An example when the input utterance is out of the capbilities of the CA.

(b) When it failed to retrieve data from knowledge graph database.

Figure 12: Two examples of the coping strategy towards problems of the conversational virtual museum guide.

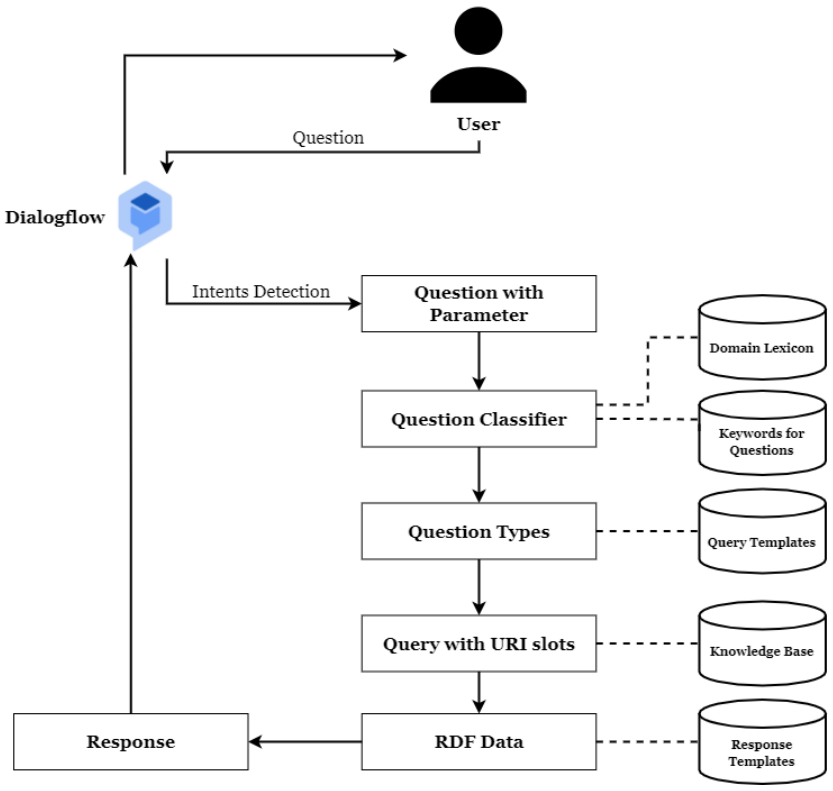


Figure 13: The design of template based KBQA component.

have. In addition, different from the chitchat conversational agents, as a conversational virtual museum guide agent focuses mainly on the art objects and exhibits, training corpora in this specific scenario are relatively scarce. Thus, in our project, building a KBQA system using machine learning methods is expensive and laborious.

In general, a KBQA system accepts natural language questions as input, builds a query based on the input question and finally gives a response by retrieving an answer using the query. In this project, the design of the template based KBQA is shown as Figure 13.

Then, we will show details about the design of each part of the KBQA component in following paragraphs.

Question Input First, during the conversation, when the user types a factoid question about entities in our knowledge graph, Dialogflow agent receives it and performs intent detection on this input utterance. If the input is recognized as the intent **Question.answer** in Table 6, the webhook service is then called and the input question along with the entities detected in it will be sent to the KBQA component in the back end.

Question Classifier Here, two dictionaries are utilized to classify the input question. The first one is Domain Lexicon, which includes the names of all entities in our knowledge graph as the lexical items. If one or more entities are matched in the question, we set these entities as subjects of the question, if no match is found here, the agent suggests the user to reform the question. The second dictionary is a set of keywords for identifying different types of questions, as Table 7 shows. With the help of these two dictionaries, the system can decide what the type and subject of the question are.

Question Types After acquiring the question type and entity list from the last step, we can translate the question to Cypher query, which we introduced in Section 2.2.2. To do so, a set of Cypher templates with slots is pre-defined. The Cypher template is selected based on the question type. A list of implemented question types can be found in Table 8.

Type	Keywords	Type	Keywords
Date	When, What time, date	Description	describe, description
Creator	Who, creator	Genre	genre, style, category
Material	material, textile	Depicts	element, object, depict, keyword
Collection	collection, museum, gallery	Exhibition	exhibition
Image	image, picture, figure	Movement	movement
Country	Which country, country, nationality	City	Which city, city

Table 7: Example for the types of keywords used to identify the question type.

Question type	**Utterance**	**Cypher query**
	Which {museum} owns <The Night Watch>?	
paintings_collection	Which {Collection} owns <Painting>?	(p:Painting)-[r:in_COLLECTION]->(?)
	Who is the {creator} of <The Night Watch>?	
paintings_creator	Who is the {Creator} of <Painting>?	(p:Painting)-[r:has_CRREATOR]->(?)
	What is the {material} of <The Night Watch>?	
paintings_material	What is the {Material} of <Painting>?	(p:Painting)-[r:on_CRREATOR]->(?)
	What is the {genre} of <The Night Watch>?	
paintings_genre	What is the {Genre} of <Painting>?	(p:Painting)-[r:belongsto_GENRE]->(?)
	What is the {key element} in <The Night Watch>?	
paintings_depicts	What is the {Depicts} in <Painting>?	(p:Painting)-[r:has_KEYWORD]->(?)
	What {movement} does <The Night Watch>belongs to?	
paintings_movement	What {Movement} does <Painting>belongs to?	(p:Painting)-[r:belongsto_MOVEMENT]->(?)
	{When} was <The Night Watch>created?	
paintings_date	{Date} was <Painting>created?	(p:Painting) return p.date
	Show me the {picture} of <The Night Watch>.	
paintings_image	Show me the {image} of <Painting>	(p:Painting) return p.image
	Which {country} is <The Night Watch>in?	
paintings_country	Which {Country} is <Painting>in?	(p:Painting)-[r:in_COUNTRY]->(?)
	Show me {paintings} in <Rijksmuseum>.	
collection_paintings	Show me {Painting} in <Collection>	(?)-[r:in_COUNTRY]->(c:Collection)

Table 8: List of the implemented question types for KBQA with examples, question type names, examples with question keywords in curly braces and entity names in angle brackets, and corresponding Cypher queries.

RDF data and Response Here, the KBQA system retrieves the answer in our cultural heritage knowledge graph using the Cypher query generated in the previous step. Then the retrieved data is matched with one of the pre-defined response templates and the slot in the template is filled. At last, the KBQA component sends the filled response template to Dialogflow agent, and the agent shows it as a response to the user in the chat box.

4.4.2 User model

The user model component aims to realize personalization for the interaction and conversation between the user and our agent. It can help the agent select information likely preferred by the user from the knowledge graph with thousands of nodes and attributes without asking the user's preference explicitly.

Also, keeping the recommendation within a non-intrusive boundary is a part of our research question, which means the agent should help the visitor focus on the exhibits and the museum visiting. As a result, we should avoid intrusive actions such as asking the visitors to fill a form for their preference. For our agent, we design the user modeling as building a user profile and modeling the user's preference by collecting the user's interaction and chat history during the conversation.

An example of the designed user profile is shown as Figure 14. When starting a conversation, the user profile is initialized to the left table in Figure 14. The initial weight to exhibit is 16 because there are in total 16 paintings in the virtual museum. During the conversation, the user profile is updated when there is an input utterance, followed by Algorithm 1.

This user profile is used to help the recommendation component give the suggestion. In the state **Entity Filter** of recommendation component, the user profile is utilized to determine which types of entity should be filtered. The probability of each interest aspect is calculated based on its weight, which is formulated as $P_i = \frac{W_i}{\sum W}$

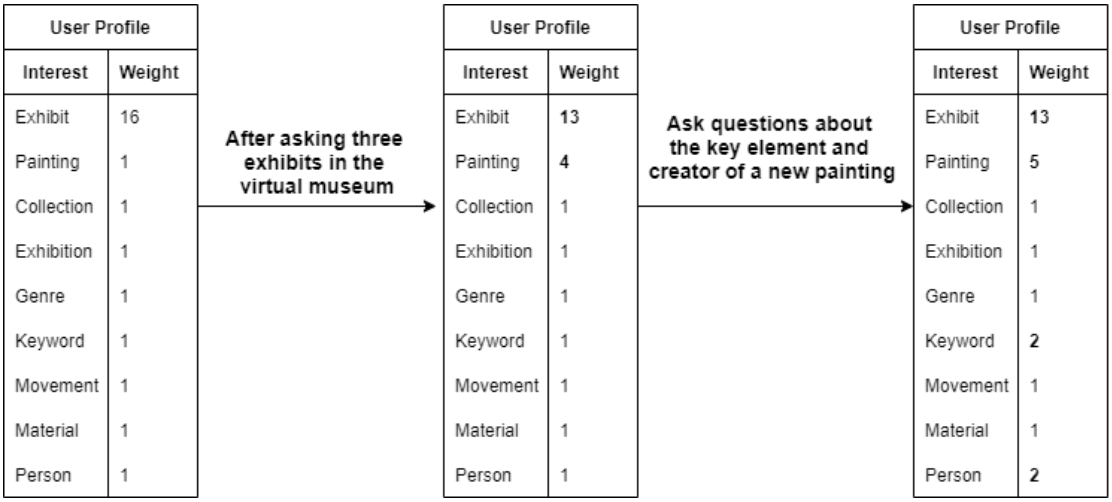


Figure 14: An example for the update of the user profile. The left table is the initial user profile.

Algorithm 1 Update user profile

Input: User's utterance U
Output: Updated user profile P_{update}
  $y \leftarrow 1$
  if Exhibit in U **then**
    $W_{Exhibit} \leftarrow W_{Exhibit} - 1$
    $W_{Painting} \leftarrow W_{Painting} + 1$
  else { Other entity except **Exhibit** in U}
    $W_{other} \leftarrow W_{other} + 1$
  end if

The case when user specifies the entity type when asking for recommendation is not included here, for instance, the user input "Show me more paintings like The Night Watch".

4.4.3 Recommendation on Knowledge Graph

Recommendation provides users with additional knowledge during the visit, helping them to explore the big knowledge graph and find exhibits of interest to themselves. As we discussed in Section 2.4.2, there are two kinds of recommendation methods, collaborative filter and content-based recommendation. In our project, though we record the user's interaction and chat history with our agent, these records are used for user modeling only during the conversation and will be deleted as the agent reboots. As no long-term user interaction history data can be used to build a collaborative filter recommendation system, content-based recommendation using embedding based method is adopted.

In our case, the design of the recommendation work flow is shown as Figure 15. We use the FastRP embedding algorithm to capture the network information and generate the node embeddings of our knowledge graph base. With the embeddings, the similarity between nodes in graph knowledge base can be calculated. As the user asks for a recommendation, our agent detects the intent of the input utterance as well as the entity mentioned, and then calls the webhook service.

Entity Filter Before calculating the similarity score, the recommendation component first does the procedure **Entity Filter**, and filters out entities within our knowledge base in two cases:

1. the user explicitly specifies the type of the entity. For example: when the input utterance is *"Please show me more paintings like The Night Watch"*, then all other types of entities except **Paintings**, such as **Person** and **Collection** in our knowledge graph, will be filtered out. If the user does not specify the type, then the component will randomly select a type, based on the probability calculation results of the **User Model**, and filter out all other types.

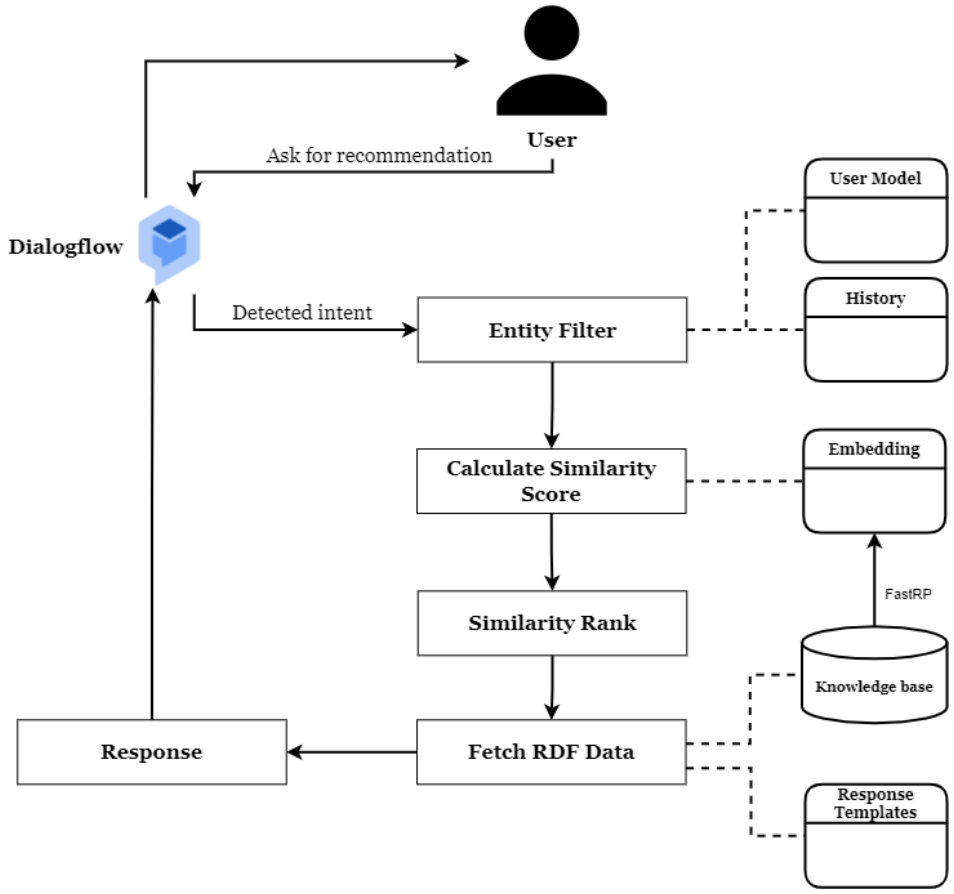


Figure 15: The design of recommendation component.

2. those entities have been recommended before. The reason for this filtering is that we want to deliver more knowledge to the user rather than show them the same artwork repetitively. Since we store the interaction and chat history between the agent and the user during the conversation, by examining the history, an entity will not be recommended twice.

Calculate Similarity Score Here, we pre-train the node embedding for our knowledge graph using FastRP and store the embedding for the calculation of this step. As to find the most similar entity, the component calculates the cosine similarity scores between the input entity and all other entities filtered.

Similarity Rank After getting the similarity score result, the component ranks the scores from highest to lowest. The entity with the highest similarity score is chosen as a recommendation to the user.

Fetch RDF Data and Generate Response With the entity selected by the last step, next the RDF data of the selected entity is fetched from the knowledge graph base. Then, the data is filled into a response template, the recommendation component sends the response back to the Dialogflow.

Connection between Entities and Visualization of Knowledge Graph Besides from suggesting a similar entity, the recommendation component can show the shared attributes between two entities and give a visualization of the knowledge graph as shown in Figure 9.

The method is demonstrated in Figure 16. To get the shared attributes between the recommendation and starting entity, the first step is to generate the Cypher query, and then, based on the retrieved path, which containing the nodes and relationships, formulate the data in a Messenger rich response

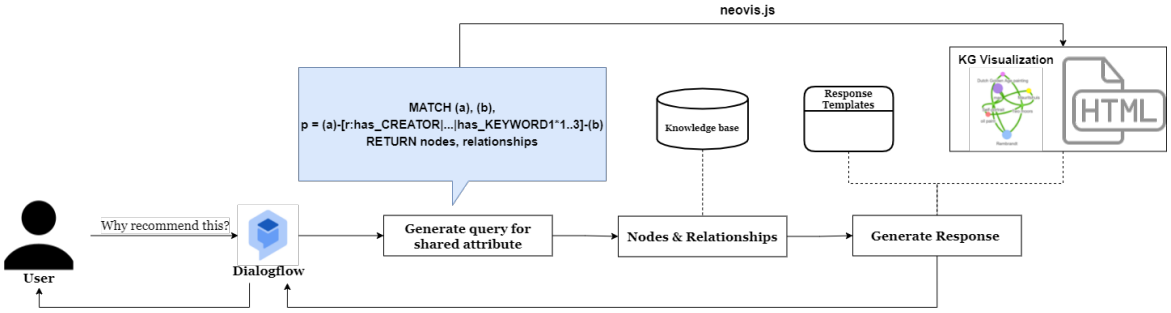


Figure 16: The design to get the shared attributes of entities and visualization of knowledge graph.

template. At the same time, we use neovis.js[23] to visualize the retrieved sub-graph and embed it in a webpage hosted on our server. And we add a button as a part of the rich response, which can open the page to view the knowledge graph visualization.

[23]https://github.com/neo4j-contrib/neovis.js/

5 User Evaluation and Result

To evaluate our knowledge graph-driven conversational virtual museum guide in this project, we used a quantitative user study as an assessment approach. In this section, we will present the design of the user evaluation, as well as results and analysis of it. In addition, this user study was performed entirely online. The candidates received the contents of the test beforehand, which includes the consent declaration, link to the online virtual museum, user scenario and evaluation questionnaire. Section 5.1 presents the procedure and design of the user study. Then the results are shown in Section 5.2, and a discussion and analysis based on the results are given in Section 5.3.

5.1 Design of the User Study

To evaluate our knowledge driven conversational virtual museum guide, we designed a series of user scenarios, see Table 9 below, and performed a user evaluation on it.

The virtual museum is divided into two rooms, as Figure 17 shows, with participants visiting the art works in each area in turn. Users have no set task in the first room; alternatively, they could view the paintings according to their interests and become familiar with the operation of the virtual museum. Table 9 describes the user scenarios for our user study, including specific types of utterances that participants are asked to have with the conversational virtual museum guide. In the second room, participants must begin talking with the virtual guide and accomplish the given assignments in the Table 9 while visiting. These tasks are related to the virtual guide's capabilities.

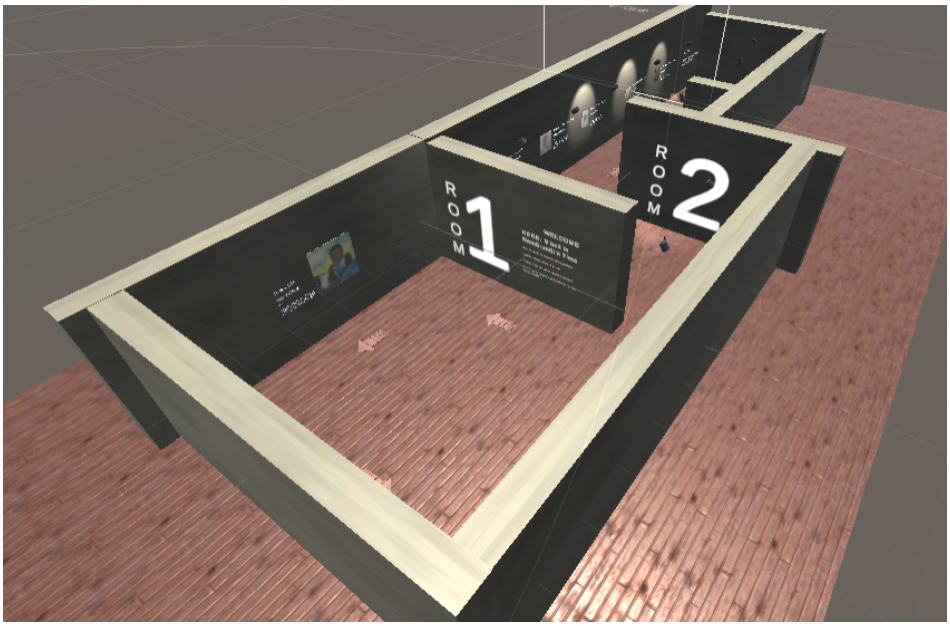


Figure 17: Two rooms in the virtual museum.

After finishing all scenarios and tasks, the participants need to fill a questionnaire based on their experience of visiting. The questionnaire is about their preference between two scenarios and their evaluation towards our conversational agent. Borsci et al. [37] present a chatbot usability scale for AI-based conversational agents. We adapted this questionnaire according to the function of our conversational virtual museum guide, and categorized the questions into usability, functionality and reliability. These three attributes follow the quality model in ISO/IEC 9126[24], and they are utilized by Niculescu et al. [38] for evaluating a touristic conversational agent.

There are in total 24 questions, and Table 10 gives a summary of the question in the questionnaire. The first 3 of them are about the user's overall opinion about the virtual guide, then 9 for usability, 8 for functionality and 4 for reliability.

The questions in the questionnaire address the *C2* and *RQ2* of this thesis, i.e. if our CA have natural dialogue with visitors. The **sub-category** in Table 10 shows the specific aspects of our CA

[24]https://en.wikipedia.org/wiki/ISO/IEC_9126

User Scenario	Question Type	Questions
1. Visit and walk through in the first room without interacting with our virtual agent	N/A	N/A
2. Move to the second room, and start having conversations with our virtual agent	Exhibition introduction	During the visiting, select at least three paintings you are interested and ask the agent for introductions about them
	Get to know the paintings by asking questions	Ask for the genre of a painting
		Ask for the key element of a painting
		Ask for the material of a painting
		Ask for which museum owns the painting
		Ask for the creator of a painting
	Explore knowledge graph by asking for recommendation	Ask for recommendation about a painting
		Ask the agent why recommend the painting after you get a recommendation
		Open the knowledge graph to see the connection between recommended entity and the starting entity

Table 9: User scenarios for user evaluation

we want to measure. Usability measures the ease of use and understandability of the museum guide, the amount of information provided in the dialogue, and the speed of response, while the functionality measures user satisfaction about intent recognition, conversation coherence, and function provided. As for reliability, questions about the agent's resilience to failure and security are asked, 2 out of 4 questions for reliability are optional, i.e. Q22 (*The agent response appropriately when it encountered a problem.*) and Q23 (*The agent helped me back to the normal conversation when a failure happened.*), which will be asked only when the participant encountered failures during the conversation.

It should be noted that **maxim of quantity** and **relevance of information** evaluate if our CA is non-intrusive during the conversation, which is related to *C2* and *RQ3* of this thesis. It suggests that our CA is non-intrusive by asking whether it offers appropriate and relevant information to the visitor, and the amount of the information is not too much or less.

And Q17 (*The agent's recommendations are always interesting to me.*) and Q20 (*My experience of visiting improved due to the recommendation from the agent.*) evaluate the performance of user model as well as recommendation by asking if the user shows interests and satisfaction on the recommendations.

For each question, the participants need to choose one of the five-level Likert scale options that best matches their feelings. For each of the five choices, from "strongly disagree" to "strongly agree", we assign a score of 1 to 5 to calculate the average score. The average score and the proportion of five options for each question are plotted in Figure 18.

5.2 Result

In total, ten volunteers participated in the user study, six of them are master students in Computer Science or Interaction Technology at the University of Twente and four are PhDs who works at the Faculty of Economics and Business of the University of Groningen. The candidates show interests and willingness in the experience of the museum visit and declare their consent to enroll the user test. They all have sufficient English reading and writing level to make conversation with the agent.

The first question is about if the participants have any experience with the museum conversational agent, the results show that 80% of them never or seldom used a conversational agent as a museum guide before. The average score of the Q2 (*Do you think your visit experience was improved by having the conversation with the virtual guide?*) is 3.8, which shows that most of them agree that the conversational virtual museum guide improved their visit experience. No one agrees that the virtual guide was annoying or spoiled the experience, 3 among them chose neutral, and 7 people chose to disagree.

The result for the usability, functionality and reliability is shown in the Figure 18, For the usability result displayed in Figure 18(a), Q4 (*I think it was easy and clear to start a conversation with the agent.*) and Q9 (*The agent gives me only the information I need.*) have the lowest score, 3.5. Many participants disagree or feel neutral to Q4, which suggests that users find it not very easy to start the

Category	Sub-category	Number	Questionnaire Item
Overall Evaluation		Q1	Have you used a chatbot or a conversational agent as a museum guide before?
		Q2	Do you think your visit experience was improved by having the conversation with the virtual guide?
		Q3	Do you think the guide was annoying and spoiled your visit experience?
Usability	Ease to start conversation	Q4	I think it was easy and clear to start a conversation with the agent.
		Q5	I would imagine that most people would learn to start the conversation quickly.
		Q6	I quickly realized what information the chatbot could give me.
	Maxim of quantity	Q7	The amount of received information was neither too much nor too less.
		Q8	The amount of recommendation from agent is appropriate.
		Q9	The agent gives me only the information I need.
	Understandability	Q10	The agent's responses were easy to understand.
		Q11	I felt comfortable with the conversational interface, i.e. the chat box.
	Speed of answer	Q12	My wait time is short for the response from the agent.
Functionality	Ability to maintain a conversation	Q13	I felt that the conversation is coherent and ongoing.
	Recognition of the user's intent	Q14	I felt my intentions were understood by the agent.
		Q15	I had to rephrase my input multiple times for the agent to be able to understand me.
	Relevance of information	Q16	During the whole conversation, the agent offers relevant information about what I asked.
		Q17	The agent's recommendations are always interesting to me.
	Satisfaction to functions	Q18	I liked to see the connections between exhibits through the knowledge graph.
		Q19	I felt asking questions to the agent was not troublesome or unnecessary to me.
		Q20	My experience of visiting improved due to the recommendation from the agent.
Reliability	Resilience to failure	Q21	Did the conversational agent have any failures during the conversation?
		Q22	The agent response appropriately when it encountered a problem.
		Q23	The agent helped me back to the normal conversation when a failure happened.
	Security	Q24	I felt my personal information is secured during the whole conversation.

Table 10: Question types and examples in questionnaire.

conversation. Based on the comments from user, it is because the icon to open the chat interface is small, and it lies in the corner of the webpage, sometimes they have to scroll the page down to see it. Besides that, the agent may not able to give sufficient guidance to help them start the conversation. As for Q9, though most users are satisfied of the information they received during the conversation, a few of them think the amount of information is too much or less. For the rest question about usability, such as the understandability and reply speed of the conversational agent, users give positive review mostly.

Figure 18(b) shows the functionality result. The description in Q15 (*I had to rephrase my input multiple times for the agent to be able to understand me.*) met half of the users' expectations and the other half did not, which means sometimes they need to rephrase the input so that their utterances can be understood by the museum guide. We learned from the feedback that when users gave up using the options provided by the agent as input and tried to enter free text instead, the situation of not being understood by the agent happened more frequently. As we mentioned in section 4.3.2, using options provided by the agent as input make the agent detect the intents more accurately, while the intent of free text can not always be perfectly recognized due to the limited training corpora for each intent. And it explains the result of Q14 (*I felt my intentions were understood by the agent.*) as well. As for Q17 (*The agent's recommendations are always interesting to me.*), around half of participants did not always show interests in the items recommended by the agent, but no one disagrees that recommendation improved their visit experience, suggesting that the recommendation component has a positive effect on the user's experience in general. In addition to the above issues, users' opinions towards other functionality questions were mostly positive.

When having a talk with the agent, four participants failed at least once, as Figure 18(c) shows, and six participants had not encountered any failure. According to user feedback, problems usually occur when users enter free text as input, and as a result, they wait interminably and are unable to receive sufficient information to help them return to the normal dialog track. As shown in Figure 18(d), the scores for Q22 (*The agent response appropriately when it encountered a problem.*) and Q23 (*The agent helped me back to the normal conversation when a failure happened.*) are 2.75 and 3, respectively, indicating that users are slightly dissatisfied with the agent's ability to withstand failure. When it comes to Q24 (*I felt my personal information is secured during the whole conversation.*), the majority of users agree that their personal information is kept private during the conversation.

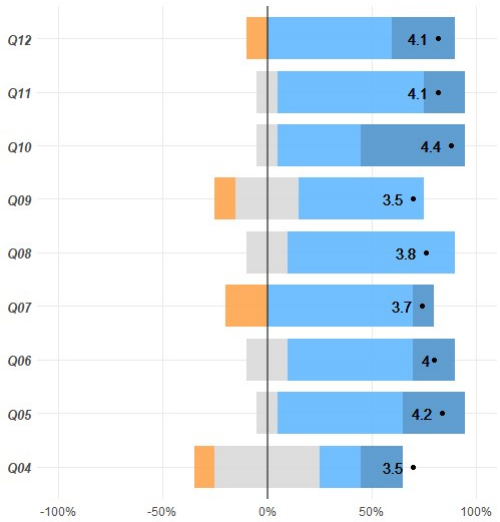
In addition to the questionnaire, the participants had comments about their experiences after having conversations with our CA, a part of the comments are shown in Table 11. The comments basically align with the results of the questionnaire, describe some potential aspects for improvement, for instance, the CA could be more flexible with free input. Few complaints about the recommendation feature and there are also positive comments towards the

More comments from user
It should be more flexible with input
I felt a bit lost on how to start, so perhaps an example interaction with the agent before starting could be useful.
It is also expected to add the audio guide during the visiting
The chat box blocking the picture, it is not easy see the painting in the virtual museum when chatting.
Background information of the painting coule be more
The bot is interesting. I am willing to visit a real museum in this way
For me, the museum logo to start the chat is a little bit hard to find
Once I waited forever for an answer, so I clicked a different option and it worked again.

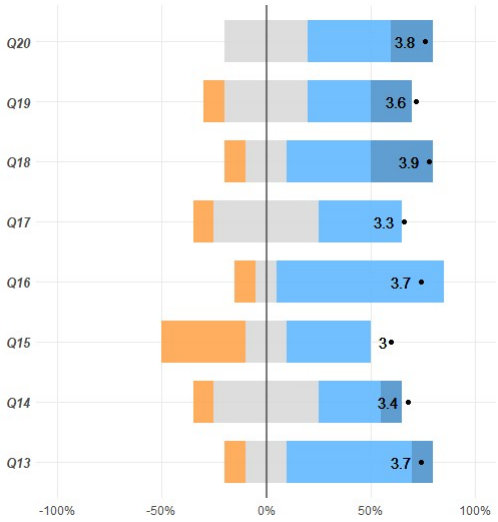
Table 11: Comments collected from the participants.

5.3 Discussion and Analysis

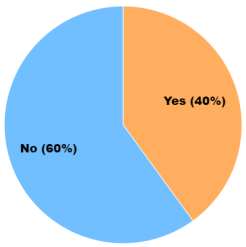
In our user evaluation, we interviewed 10 participants, asked them to visit an online exhibition in a virtual museum and complete a questionnaire containing 24 questions. Because CA is a new function



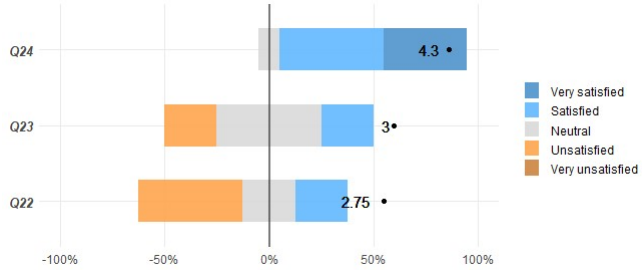
(a) Result on usability (Q04 to Q12)



(b) Result on functionality (Q13 to Q20)



(c) Result of Q21: Did the conversational agent have any failuresduring the conversation?



(d) Result on reliability (Q22 and Q23 were asked only when answer of Q21 was "Yes")

Figure 18: Results of user evaluation on usability, functionality and reliability

for the museum, and the thesis time is limited, we decided to interview 10 people as an initial user review. The feedback from ten individuals should be enough to figure out the CA's major issues. And a user test with a larger scope can be performed in the future for more precise feedback. What's more, to make the evaluation more reliable, we showed a user scenario and instruction for participants to make sure they can experience all features of the museum CA. Not only results of the questionnaire, but also free comments about the agent and virtual museum were collected as shown in Table 11. Thus, we believe that the user study results are useful for identifying current issues and offering insights in future research.

The findings in Section 5.2 show that our conversational agent improves users' overall visit experience by assisting them in exploring cultural assets in the knowledge graph. Users generally approve of the usability of our museum conversation agent, indicating that it performs well in terms of ease of use, reaction time, amount of information offered, and intelligible response. Users are happy with the majority of functions, such as the consistency of the dialogue, the relevancy of the information offered, the display of the knowledge graph, Q&A and the recommendation function, according to the functional survey results.

At the same time, the survey findings emphasize the issue of our museum conversation agent's unstable detection of the intent of free text input, as well as its silence and limited guidance when failing to interpret user input. Due to the limited performance in current, here the functionalities still need to be upgraded or changed, besides our design for the error handling in Section 4.3.3,

Furthermore, our validation method has limits. First and foremost, while the size of our population is enough to identify the major issues with the museum CA's current function, as visitors all followed the tasks, this relatively small sample size is of limited use in improving our dialog flow, since the variety of target users can not be included in a size of 10 population. Through long-term and continuing user testing, such as Maximo [34], which had 7,000 conversations in iterative user testing to perfect its conversational skills, we can improve the conversational capabilities of museum CA in the future. Second, our survey simply asks about general satisfaction with the Q&A and suggestion functionalities, a more specific user evaluation can be performed over the functions, or implementing like and dislike for the recommendation and answers of the question, to collect the user's feedback during the conversation. Last but not least, in our thesis we use ISO/IEC 9126, an international standard for the evaluation of software quality, as the guideline to design our questionnaire, but a recently published survey [1] about museum chatbots suggested the user evaluation questions to align qualitative attributes in ISO 9241, which covering ergonomics of human-computer interaction.

6 Conclusion

6.1 Overview of the Approach

In this thesis, we answered our research questions by proposing a knowledge graph-driven conversational virtual museum guide to assist visitors in exploring art exhibitions using natural language text dialogues, as well as to share knowledge with users via knowledge graph-based Q&A and recommendations to improve the user experience of visiting museums.

First, we looked at present conversational bots, particularly museum conversational agents, in the literature review section. We found that although chattering bots have made significant progress, museum conversational agent research is relatively limited, especially on leveraging the knowledge graph. Prior virtual museum guides have always had an appealing appearance or narrative as a part of the museum, and they focus on leveraging multimedia to engage visitors in conversation and provide basic information about the collections and museums. They seldom utilize knowledge graphs to enrich their response. What's more, basically none of them leverage a knowledge graph to answer factoid questions beyond the questions set and provide recommendations. As a result, we were motivated to design a conversational virtual museum guide driven by knowledge graph.

For the first research question, how to collect a cultural heritage knowledge graph for conversational virtual museum agent in our thesis, we found cultural heritage knowledge graphs now can be obtained from public data sources such as Wikidata thanks to the advancement of LOD. Around 5,000 entities and 30,000 relationships were extracted from Wikidata using the SPARQL query service and, then stored as our Neo4j knowledge graph database.

Second is how to build a knowledge graph driven conversational virtual museum guide which could have natural dialogue with visitors. After constructing the knowledge base, we worked on the framework for building the conversational agent, i.e. Dialogflow in our case. It simplifies the process of building and integrating by offering a user-friendly interface, and the built-in ML and NLU components ease the process to build features for our CA such as intent detection. After investigating the state-of-the-art museum chatbots, we found that knowledge graph technology is seldom utilized in CAs. Then, we listed the requirements of the knowledge graph driven conversational museum agent, designed and implemented the dialog flow of our conversational virtual museum guide, which can guide the user to explore the virtual exhibition and offering multimedia such as figures, links and suggestion chips.

Our third research question is how should we share knowledge with visitors through our conversational agent for user's interest and being non-intrusive? As a rarely explored requirement for current state of the art, sharing knowledge in our museum CA is implemented through two features, KBQ&A and personalized recommendation. They are created as webhook service and can be called by the agent when needed. The KBQ&A component utilized a template-based method, which translates the user's input text as Cypher language to query in the knowledge graph database, and fills the query result in a response template as the reply. The personalized recommendation is built based on knowledge graph embedding method and user model, and gives recommendations by calculating and ranking cosine similarity scores using node embedding matrix. The user model records the user's preference through the conversation without bothering the user, which makes the museum CA non-intrusive.

To evaluate our design of the knowledge graph driven conversational virtual museum guide, We performed a user study and 10 participants were involved. There were two stages to the evaluation process. Participants visited a web virtual museum in the first phase and interacted with the conversational virtual museum guide throughout their stay. They were then invited to complete questionnaires to rate the performance of the museum's virtual guides in the second phase. The questionnaire was adapted based on the chatbot usability scale by Borsci et al. [37], and we categorized the questions into usability, functionality and reliability according to the quality attributes introduced in ISO/IEC 9126.

6.2 Main Lessons Learned and Limitations

According to the user evaluation result, overall, most users agreed that the conversational virtual museum guide improved their experience. The results for the agent's usability and functionality look very promising, and skills of the agent, particularly recommendation, are favorably evaluated. However, we have also received feedback that the conversational agent occasionally fails to respond to users, as well as insufficiency in the resilience to the failure.

Apart from the evaluation results, during the study and experiments for the thesis, we also learned that, for example, the museum chatbots benefit from KGs' rich structured machine-understandable information, which solves the problem of limited and unstructured knowledge sources. But, as Savvas et al. [1] mentioned, the quality of the knowledge graph is critical to the success of the proposed approach and the performance of the CA. With the quality issue of the knowledge graph handled, knowledge graph-based features in museum CA, such as the Q&A and recommendation we presented and other possible features, are promising.

Another thought is that having multiple rounds of user testing would be helpful to build a natural and robust CA for museum. Currently, our museum CA tries to take a proactive position during the conversation, which guides the users to follow the path by providing options or examples. According to the reviews from the users, this proactive way helps the user not to feel lost through the exploring the museum, but not all the visitors are interested in this way. our museum CA has shortcomings, such as being limited when interpreting free-text input. Having more user testing, or get more users involved in the evaluation, can collect much more useful feedbacks to improve it.

Besides that, the issues and limitations that exist in the current edition of the conversational museum virtual guide need to be addressed. For example, some users find that it is not very easy to start the conversation due to the insufficiency of the guidance. To solve it, more examples, as well as a detailed instruction, can be added to help the user understand how the CA works. Another one is that users are more likely to employ the options during the conversation because the agent can not understand some free text questions as correctly as when dealing with template options. Besides that, the ability to recover from failures should be enhanced. When problems occur or the agent is unable to interpret the input, it should provide information or guidance to assist users get back on track.

6.3 Future work

As a relatively underexplored field, conversational virtual museum guides based on knowledge graphs have lots of potentials. The following aspects of our work can be enhanced in the future.

First, while Wikidata has the advantage that, as a public data resource, it contains a huge number of entities more than others and can be edited by users, there are some current issues that should be noted in the study of Dadalto et al. [39]. When we extracted the knowledge graph from Wikidata as knowledge base of cultural heritage, attributes about taxonomies such as **subclass of** were not involved much, since we only needed a small group of entity types to build the knowledge graph. However, the stratification of taxonomies in Wikidata is not well executed [39]. Besides that, the phenomenon of polysemy in Wikidata can not be neglected, and millions of non-professional users, creating the entities and referencing each other makes the problem even worse. Thus, if the knowledge graph needs to be enlarged, we must consider how to eliminate the issues in Wikidata, or changing to another LOD data source, such as Europeana or DBpedia, is also possible.

In addition, current features of our conversational agent can be improved in different ways. For instance, using advanced NLP capabilities to generate more natural descriptions based on the query result from knowledge graph. The current CA cannot disambiguate, which is an important feature to help reduce errors when operating on entities with the same name that exist in the knowledge base. Furthermore, adopting a different knowledge graph embedding method, which may preserve more graph information, or changing the method for the recommendation component to collaborative filter could improve the performance of the agent. It can also deal with the cold-start issue in our agent. And more rounds of user evaluation can be considered. We can also adapt the design of our current user evaluation and questionnaire with ISO 9241, covering ergonomics of human-computer interaction.

References

[1] Savvas Varitimiadis, Konstantinos Kotis, Dimitra Pittou, and Georgios Konstantakis. Graph-based conversational ai: Towards a distributed and collaborative multi-chatbot approach for museums. *Applied Sciences*, 11(19):9160, 2021.

[2] Michael Frederick McTear, Zoraida Callejas, and David Griol. *The conversational interface*, volume 6. Springer, 2016.

[3] Thomas Pellissier Tanon, Marcos Dias de Assuncao, Eddy Caron, and Fabian M Suchanek. Demoing platypus–a multilingual question answering platform for wikidata. In *European Semantic Web Conference*, pages 111–116. Springer, 2018.

[4] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems*, pages 2942–2951, 2018.

[5] Nhuan D To and Marek Reformat. Question-answering system with linguistic terms over rdf knowledge graphs. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4236–4243. IEEE, 2020.

[6] Joan Plepi, Endri Kacupaj, Kuldeep Singh, Harsh Thakkar, and Jens Lehmann. Context transformer with stacked pointer networks for conversational question answering over knowledge graphs. *arXiv preprint arXiv:2103.07766*, 2021.

[7] Nadine Steinmetz, Bhavya Senthil-Kumar, and Kai-Uwe Sattler. Conversational question answering using a shift of context. In *EDBT/ICDT Workshops*, 2021.

[8] Tsvi Kuflik and Cesare Rocchi. User modelling and adaptation for a museum visitors' guide. In *PEACH-Intelligent Interfaces for Museum Visits*, pages 121–144. Springer, 2007.

[9] Cynthia A Thompson, Mehmet H Goker, and Pat Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004.

[10] Dietmar Jannach. Advisor suite - a knowledge-based sales advisory system. In *Proceedings of the 16th European Conference on Artificial Intelligence*, ECAI'04, page 720–724, NLD, 2004. IOS Press.

[11] Arpit Rana and Derek Bridge. Navigation-by-preference: A new conversational recommender with preference-based feedback. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, page 155–165, New York, NY, USA, 2020. Association for Computing Machinery.

[12] Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. Improving the user experience with a conversational recommender system. In *International Conference of the Italian Association for Artificial Intelligence*, pages 528–538. Springer, 2018.

[13] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer, 2015.

[14] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems, 2020.

[15] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. 29(12):2724–2743. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[16] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795. Curran Associates Inc.

[17] Meihong Wang, Linling Qiu, and Xiaoli Wang. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3):485, 2021.

[18] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. Rdf2vec: Rdf graph embeddings and their applications. *Semantic Web*, 10(4):721–752, 2019.

[19] Finn Årup Nielsen. Wembedder: Wikidata entity embedding web service. *arXiv preprint arXiv:1710.04099*, 2017.

[20] Jan Portisch, Michael Hladik, and Heiko Paulheim. KGvec2go – knowledge graph embeddings as a service. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5641–5647, Marseille, France, May 2020. European Language Resources Association.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[22] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. Fast and accurate network embeddings via very sparse random projection, 2019.

[23] Ioannis Papaioannou, Amanda Cercas Curry, Jose L Part, Igor Shalyminov, Xinnuo Xu, Yanchao Yu, Ondrej Dušek, Verena Rieser, and Oliver Lemon. Alana: Social dialogue using an ensemble model and a ranker trained on user feedback. *Alexa Prize Proceedings*, 2017.

[24] Amanda Cercas Curry, Ioannis Papaioannou, Alessandro Suglia, Shubham Agarwal, Igor Shalyminov, Xinnuo Xu, Ondřej Dušek, Arash Eshghi, Ioannis Konstas, Verena Rieser, et al. Alana v2: Entertaining and informative open-domain social dialogue using ontologies and entity linking. *Alexa Prize Proceedings*, 2018.

[25] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93, 2020.

[26] P. de Almeida and Y. Shigeo. Interactive conversational character as a virtual tour guide to an online museum exhibition. In *International Conference on Computers in Education, 2002. Proceedings.*, pages 215–216 vol.1, 2002.

[27] Stefan Kopp, Lars Gesellensetter, Nicole C Krämer, and Ipke Wachsmuth. A conversational agent as museum guide–design and evaluation of a real-world application. In *International workshop on intelligent virtual agents*, pages 329–343. Springer, 2005.

[28] William Swartout, David Traum, Ron Artstein, Dan Noren, Paul Debevec, Kerry Bronnenkant, Josh Williams, Anton Leuski, Shrikanth Narayanan, Diane Piepol, Chad Lane, Jacquelyn Morie, Priti Aggarwal, Matt Liewer, Jen-Yuan Chiang, Jillian Gerten, Selina Chu, and Kyle White. Ada and grace: Toward realistic and engaging virtual museum guides. In Jan Allbeck, Norman Badler, Timothy Bickmore, Catherine Pelachaud, and Alla Safonova, editors, *Intelligent Virtual Agents*, Lecture Notes in Computer Science, pages 286–300. Springer.

[29] Timothy W Bickmore, Laura M Pfeifer Vardoulakis, and Daniel Schulman. Tinker: a relational agent museum guide. *Autonomous agents and multi-agent systems*, 27(2):254–276, 2013.

[30] Stavros Vassos, Eirini Malliaraki, Federica dal Falco, Jessica Di Maggio, Manlio Massimetti, Maria Giulia Nocentini, and Angela Testa. Art-bots: Toward chat-based conversational experiences in museums. In Frank Nack and Andrew S. Gordon, editors, *Interactive Storytelling*, Lecture Notes in Computer Science, pages 433–437. Springer International Publishing.

[31] Octavian-Mihai Machidon, Aleš Tavčar, Matjaž Gams, and Mihai Duguleană. Culturalerica: A conversational agent improving the exploration of european cultural heritage. *Journal of Cultural Heritage*, 41:152–165, 2020.

[32] Multilingual europeana chatbot. `https://culturebot.eu/`. Accessed: 2021-11-16.

[33] Cartier foundation for contemporary art bot. `https://www.fondationcartier.com/en/`. Accessed: 2021-11-16.

[34] MÁximo the titanosaur. https://www.fieldmuseum.org/exhibitions/maximo-titanosaur?chat=open. Accessed: 2021-11-16.

[35] Stefan Kopp, Lars Gesellensetter, Nicole C. Krämer, and Ipke Wachsmuth. A conversational agent as museum guide – design and evaluation of a real-world application. In Themis Panayiotopoulos, Jonathan Gratch, Ruth Aylett, Daniel Ballin, Patrick Olivier, and Thomas Rist, editors, *Intelligent Virtual Agents*, Lecture Notes in Computer Science, pages 329–343. Springer.

[36] Culture chatbot: a virtual museum tour guide. https://rasa.com/showcase/jhn-ams/. Accessed: 2021-11-16.

[37] Simone Borsci, Alessio Malizia, Martin Schmettow, Frank Van Der Velde, Gunay Tariverdiyeva, Divyaa Balaji, and Alan Chamberlain. The chatbot usability scale: the design and pilot of a usability scale for interaction with ai-based conversational agents. *Personal and Ubiquitous Computing*, pages 1–25, 2021.

[38] Andreea I Niculescu, Kheng Hui Yeo, Luis F D'Haro, Seokhwan Kim, Ridong Jiang, and Rafael E Banchs. Design and evaluation of a conversational agent for the touristic domain. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–10. IEEE, 2014.

[39] Atílio A Dadalto, João Paulo A Almeida, Claudenir M Fonseca, and Giancarlo Guizzardi. Type or individual? evidence of large-scale conceptual disarray in wikidata. In *International Conference on Conceptual Modeling*, pages 367–377. Springer, 2021.