DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Using Knowledge Graphs to Improve News Search and Exploration with Voice-Based Conversational Agents

Nils Justus Rehtanz

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Using Knowledge Graphs to Improve News Search and Exploration with Voice-Based Conversational Agents

Verwendung von Wissensgraphen zur Verbesserung der Nachrichtensuche und -erforschung mit sprachbasierten Gesprächsagenten

Author:	Nils Justus Rehtanz
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	M.Sc. Phillip Schneider
Submission Date:	15.05.2023

I confirm that this bachelor's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.05.2023 Nils Justus Rehtanz

Acknowledgments

I would like to take this chance to express my sincere gratitude for the constant support and advice of my thesis advisor M.Sc. Phillip Schneider. He provided me with the necessary guidance and expertise in this field and has helped me pinpoint the most important aspects. Additionally, I express my sincere gratitude to him for introducing me to the fast-evolving and vitally important world of knowledge graphs and conversational agents.

Abstract

Knowledge graph driven conversational agents in general as well as knowledge graphs in particular for news article related tasks have been studied in the past separately. However, their combination, i.e., knowledge graph driven conversational agents for news exploration, seems to be an underinvestigated area. This thesis addresses how to create a knowledge graph for a voice-based conversational agent for news exploration and how to implement the agent to use this knowledge graph for news search and recommendation. For this purpose, the current state-of-the-art in voice-based news search and exploration is analyzed, and the essential potentials for improvement are identified. Furthermore, interaction patterns for voice-based German news search and exploration are evaluated, serving as a basis for the knowledge graph and the conversational agent. On this foundation, the knowledge graph is conceptualized and constructed, and the conversational agent is developed. User tests have been carried out to gain insights into how to further improve the system in future work.

Keywords: Voice-based conversational agent, natural language processing, graph-based database models, search interfaces, news search

Kurzfassung

Grundsätzliche Wissensgraph-gesteuerte Gesprächsagenten sowie Wissensgraphen spezifisch angewendet auf Nachrichtenartikel wurden in der Vergangenheit einzeln untersucht. Ihre Kombination, also Wissensgraph-gesteuerte Gesprächsagenten für die Nachrichtenrecherche, scheint jedoch ein unterforschtes Gebiet zu sein. Diese Arbeit behandelt, wie ein Wissensgraph für einen sprachbasierten Gesprächsagenten für die Nachrichtenrecherche erstellt wird und wie der Agent implementiert wird, um diesen Wissensgraphen für die Nachrichtensuche und -empfehlung zu nutzen. Zu diesem Zweck wird der aktuelle Stand der Technik in der sprachbasierten Nachrichtensuche und -recherche analysiert und die wesentlichen Verbesserungspotenziale identifiziert. Darüber hinaus werden Interaktionsmuster für die sprachbasierte Nachrichtensuche und -recherche in Deutsch evaluiert, die als Grundlage für das Konzept des Wissensgraphen und des Konversationsagenten dienen. Auf dieser Grundlage wird der Wissensgraph konzipiert und konstruiert sowie der Gesprächsagent entwickelt. Benutzertests werden durchgeführt, um Einblicke zu gewinnen, wie das System in zukünftigen Arbeiten weiter verbessert werden kann.

Schlüsselwörter: Sprachbasierte Gesprächsagenten, Natural Language Processing, graphbasierte Datenbankmodelle, Suchschnittstellen, Nachrichtensuche

Contents

1. Introduction

1.1. Motivation

The interaction between humans and computers has drastically changed over time [1]. Starting with the operation of computers via terminals, later with a mouse and graphical user interface, and eventually through touch. Nowadays, people speak or use gestures to interact with their smartphones, smart speakers, or even their cars [2]. The rapid development of artificial intelligence and systems such as Mercedes-Benz's MBUX[1] or assistants like Amazon Alexa[2], Apple Siri[3], and Google Assistant[4] enable increasingly natural interactions. Given their ease of use and greater user experience, these assistants enjoy widespread popularity [3].

Developments like ChatGPT[5], a large language model [4] are a driving force behind the paradigm shift in human-computer interaction toward the medium of language. This development illustrates that people want to interact with a computer in a natural way. As a result, dialogue systems, also known as conversational agents (CAs), are a long-established research area in the field of natural language processing (NLP) [5]. They have evolved over a long period of time and benefit from advances in neural network-based methods. Consequently, they are now widely used in many areas, such as customer service and healthcare [6, 7, 8].

Voice interfaces offer significant advantages: they are faster, more intuitive, adaptive, and better suited for people suffering from visual impairments [9]. Although most dialogue systems are currently designed to achieve a specific task or perform social chit-chat [10], there is a research trend to deploy CAs for information retrieval [11]. Compared to traditional systems that only allow single responses, CAs enable multi-turn dialogues for information retrieval. Therefore, the use of CAs for news search and exploration shall be investigated in this bachelor's thesis. In the exploratory search environment of news, where users often have only a vague idea of what they are looking for because they do not know precisely what is happening in the world, such a system is optimally suited. Users can formulate meaningful queries without prior knowledge, as they can iteratively adjust their search queries. This intuitive, dialogue-based interaction helps address issues like vocabulary mismatch [12].

[1]MBUX: https://www.mercedes-benz.com.my/passengercars/mercedes-benz-cars/mbux.html
[2]Amazon Alexa: https://developer.amazon.com/en-US/alexa
[3]Apple Siri: https://www.apple.com/siri/
[4]Google Assistant: https://assistant.google.com/
[5]ChatGPT: https://openai.com/blog/chatgpt

Knowledge graphs (KGs) have often proven to be extremely suitable as a data structure for such and many other NLP tasks, as they capture the interconnected nature of domain-specific knowledge [5]. Due to their rich representation of entities and semantic relationships, they have frequently been investigated to improve dialogue systems by facilitating contextual utterance understanding or retrieval-based response generation [5]. These are just a few reasons why KGs help support the disambiguation of complex search queries, where the meaning of expressions can vary depending on the context. In addition, KGs allow users to explore further concepts and facets of a topic by easily leveraging the relationships between individual pieces of information [13]. In this context, the combination of KGs and CAs for news exploration offers a promising research direction, which seems to be under-investigated so far.

In this bachelor's thesis, we explore how to create a KG for a voice-based CA for news exploration and how the agent can use this KG for news search and recommendation. By combining the capabilities of dialogue systems with the structured representation of information provided by KGs, we aim to enhance the user experience and the overall effectiveness of the system. Through user tests, we aim to gain insights to improve the system in future work further, thus contributing to the development of more natural and effective human-computer interaction in the field of news search and exploration.

1.2. Research Questions

CAs and their use cases are burgeoning technology and consequently, research is currently taking place and evolving fast. In fact, a CA focusing specifically on German news search and exploration has not yet been conceived or tested. In addition, as of March 2023, no research papers regarding this topic have been published to our knowledge. Therefore, this clear gap in the research of CAs warrants this thesis' main research goal:

How can we use a knowledge graph to develop a conversational agent for German news search and exploration?

This bachelor's thesis deals with determining insights into the current state of CAs used for news search. It presents how we developed a prototype. It analyzes the first users' feedback from an initial user test. For this purpose, we have formulated five research questions.

RQ1: *What is the current state-of-the-art in voice-based news search and exploration?*

One of our goals is to determine the status quo of the most popular systems, such as Apple Siri or Amazon Alexa, in the area of news search. In addition, we are investigating which approaches are already being pursued in academia. To this end, insights relevant to this thesis' prototype will be extracted from published user studies and research papers covering voice-based news search.

RQ2: *What are suitable interaction patterns for voice-based German news search and exploration?*

Since such a system has never been developed before, it is of paramount importance to first explore how users interact with the system. The consequent discoveries of interaction patterns will form the basis to successfully create a voice-based CA.

RQ3: *How to construct a German news knowledge graph as the database for a conversational agent?*

To answer research question 3 we will address how to structure the German news KG, which data we use, and how we develop it in accordance with the agent's needs.

RQ4: *How to build a voice-based conversational agent for news search and exploration with the knowledge graph?*

This research question evolves around using the findings of the previous research question and determining how the developed KG can be used to build the CA. Here we explore the architecture of the agent and how we develop it.

RQ5: *Which insights can be gained from user tests for improving the conversational agent?*

Finally, we will clarify how the previous agent is perceived by users and what insights we can draw from the user tests in order to further develop and improve the agent in the future.

1.3. Structure of this Thesis

The first chapter outlines the motivation for this thesis, the research questions, and the structure. Chapter 2 describes the theoretical background necessary for this thesis. The information gathered will form the basis for the development of the prototype. Topics of interest include conversational dialogue systems, KGs, and why they should be used in tandem. The subsequent chapter 3 deals with all current papers related to KGs and dialogue systems. Additionally, user studies in the area of news search are analyzed. Chapter 4 illustrates the methodology of developing the prototype including specific steps. The following chapter 5 describes the relevant results for the presented research questions. This includes how we developed the agent and an analysis of the user study. Chapter 6 discusses and interprets the previously discovered findings and shows the agent's limitations. The final chapter 7 details this thesis' conclusion and gives a direction for future work.

2. Theoretical Background

In this chapter, we will explain the concepts and theoretical foundations necessary for the development of this thesis prototype. At first, a general dialogue system will be defined and existing frameworks used to create a dialogue system will be demonstrated. Afterward, the concept of knowledge graphs (KGs) will be elaborated in detail, since we use it as a database for our system. At last, we will formulate the reasons to connect a dialogue system with a KG and the inherent advantages of this combination.

2.1. Dialogue systems

2.1.1. Definition

Dialogue systems are computer programs designed to interact with human users in a natural, human-like manner through speech, text, or a combination of both modalities [14, 15]. They have gained significant attention in recent years due to advancements in NLP, machine learning, and artificial intelligence, which have led to the development of more sophisticated and human-like systems [16, 17]. Dialogue systems are employed in various applications, such as customer support, personal assistants, recommendation systems, and tutoring, with the primary goal of facilitating user interactions with computer systems and providing assistance, information, or performing specific tasks on the user's behalf [10, 18].

Such systems typically consist of the following components: input decoder, natural language understanding (NLU), dialogue manager (DM), domain specific component, response generator, and output renderer [14], as depicted in Figure 2.1. The input decoder is the component that recognizes the user input. However, this component can only be found in non-text-based dialogue systems. During recognition, the user utterance is translated into text. In addition to spoken input, other inputs such as gestures or handwriting can also be processed. The component responsible for understanding what the user wants to express is the NLU. It converts the words into a semantic representation. After the keywords and their meaning are figured out, they are passed to the DM, which manages all aspects of the dialogue. Then, the DM takes the semantic representation of the user's text received from the NLU and determines the semantic response of the system by placing the response in the general context of the dialogue. Besides the classification into the dialogue context, the DM does not only store the entire dialogue history and fetch different data from several sources, but it also decides on a semantic answer and constructs it accordingly. In order to cope with these versatile tasks in a dialogue system, it consists of different components including the

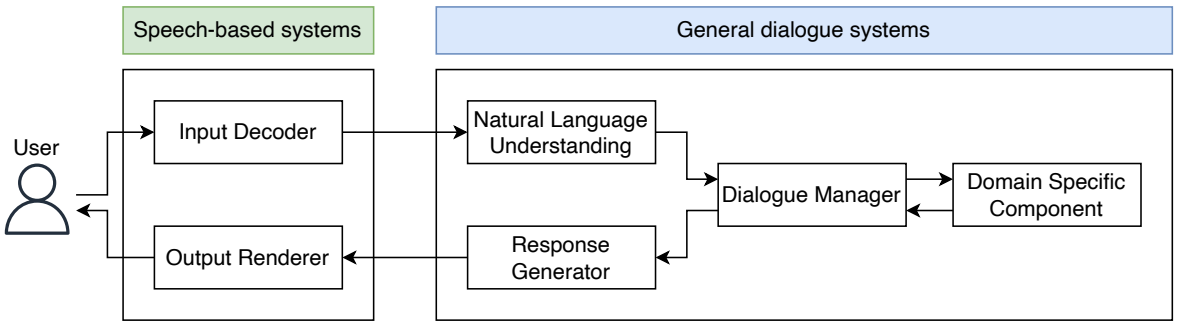


Figure 2.1.: Architecture of general dialogue systems

dialogue model, the user model, and the knowledge base.

As the DM may have to interact with external software components, a dialogue system also consists of domain-specific components. These components have the task of constructing the SQL database queries from the information contained in the DM, for example. The response generator then takes the semantic response from the DM, structures the information received, and determines the choice of words, as well as the syntactic structure of the message. As an example, the information from the semantic response is simply inserted into already prefabricated templates. If it is a voice-based system, the dialogue system also has an output renderer. The output renderer uses text-to-speech to convert the response into speech so that it can be played back to the user.

Despite the aforementioned rapid development in the areas of NLP, machine learning and artificial intelligence, there are still many challenges and problems in the development of dialogue systems. Firstly, NLU is problematic as it is difficult to deal with ambiguities, variations and complexity in user input [19]. Furthermore, it is complicated to extract relevant information from unstructured text or data [20]. As such, dialogue management is an intricate task because it is difficult to present and maintain the context of a conversation [21]. Moreover, it is equally complicated for a dialogue system to decide on a suitable answer based on the user input and the dialogue history [20]. Another challenge is the so-called context awareness, which, for example, means the adaptation of the system's behavior to the preferences and needs of the user [19]. Further, tracking and updating the dialogue status to maintain conversational coherence is difficult [21]. The disambiguation problem refers to the processing of ambiguity in user input [21] or also to the identification of the most relevant information based on user input and context [20]. In addition, given the complexity of dealing with a wide range of user inputs, intents, and topics in different areas [20], dialogue systems have the problem of not being easily scalable and adaptable. It is difficult for a dialogue system to be extended by domains or tasks without requiring substantial retraining [19]. Another challenge is to evaluate the system methodically and to find suitable evaluation metrics for dialogue systems [22]. For this purpose, criteria for the quality of system responses in terms of relevance, informativeness, and naturalness, which cannot always be evaluated

unambiguously, are often used [17].

In recent years, the development of dialogue systems has evolved significantly with many breakthroughs marking the transition from early rule-based systems to more advanced data-driven and neural approaches. ELIZA is one of the first and most well-known examples of a dialogue system being already developed in the mid-1960s [23]. It used pattern matching and substitution techniques to replicate human-like conversation [23]. Thereafter, dialogue systems evolved from the early rule-based approach to systems that understand both syntaxes as well as semantics, such as the SHRDLU system. This system demonstrates NLU by allowing users to interact with a simulated block world through typed commands [24]. Later, during the late 1990s and early 2000s, statistical and data-driven models were implemented with the goal of optimizing dialogue systems [25]. Thereafter, task-oriented dialogue systems, which help users to fulfill specific tasks through natural language interaction, followed [20]. Next, neural and deep learning approaches, like sequence-to-sequence models, and attention mechanisms to improve NLU and response generation of dialogue systems, were introduced [26, 27]. The newest developments in the area of dialogue systems are open-domain and knowledge-grounded systems. These dialogue systems, for example, OpenAI ChatGPT, were developed to have general conversations with users about various topics [28].

Dialogue systems have found widespread use across numerous domains. Virtual assistants like Apple Siri, Amazon Alexa, and Google Assistant are popular examples of dialogue systems, which support users to perform multiple tasks and answer questions. Moreover, dialogue systems are being used for customer service as they help firms automate their customer service processes by providing quick responses to customer inquiries [8]. Another application domain for dialogue systems is in e-commerce where they are primarily being used to support customers, for example, in the ordering process and other purchasing-related tasks [29]. Furthermore, they are helpful in the health and well-being sector since dialogue systems can be used as mental health chatbots that can provide emotional support, for example [7]. Another use case of dialogue systems is in the education domain where they are deployed as virtual tutors [30].

2.1.2. Types of dialogue systems

Dialogue systems can be distinguished into two categories, namely task-oriented and non-task-oriented, also known as open-domain, systems. In this section, the two categories are differentiated further.

First, task-oriented dialogue systems are designed for goal-driven interactions [25]. This means that these systems were designed to support users perform certain tasks in specific areas [31]. Since task-oriented dialogue systems have a specified target and consequently only interact in a particular area, they provide users with more accurate and focused assistance [32]. For this aim, these systems are often built on an ontology, which contains all the important information about the domain in which the system interacts [31]. For example,

for a pizza ordering system, the information about this domain would be available payment methods, pizza sizes, and toppings. These systems are not just about having a coherent conversation, but these systems are more concerned with guiding the user in the conversation toward a specific goal [33]. Task-oriented dialogue systems often employ a frame-based approach, using predefined templates or so-called "frames" to gather information and guide the conversation, since this is best done in case of a clear conversation flow [34]. In addition, in order to fulfill the task in a targeted manner, these systems conduct mixed-initiative conversations with the user [35]. This means that both the user as well as the system can take control of the conversation. However, the developer of the system can decide whether the system is system-driven or user-driver or a combination of both [36]. Additionally, it is of paramount importance that these systems can effectively handle errors and recover from misunderstandings to ensure user satisfaction and task completion [37]. Regarding the previously described pizza ordering example, it would be problematic if only half of the pizza ordering process would be executed. These systems often use dialogue state tracking to store the context of the dialogue to prevent such a half-order scenario [38].

Dialogue systems can also enable a multimodal interaction [39]. As such, task-oriented systems enable the combination of speech, text and visual elements to enhance the user experience and task completion. To evaluate such a system, metrics such as task completion rate, user satisfaction and dialogue efficiency are often used [40]. Examples of task-oriented dialogue systems are systems for ordering pizza, booking flights or hotel rooms, or even a virtual tour guide assistant.

The second category of dialogue systems called non-task-oriented dialogue systems or also open-domain dialogue systems do not have a defined goal since they aim to engage the user in general conversations without focusing on specific tasks or goals [41, 42]. Therefore, these systems are designed to cover various topics and can demonstrate information regarding different subject areas [43]. Non-task-oriented dialogue systems often focus on simulating emotions and personality traits to create a more human-like interaction experience for users in order to engage a user in the conversation [44]. However, this is particularly difficult to develop as there are many challenges in enhancing the human-like, natural conversation capabilities of the system. Such difficulties include maintaining coherence, dealing with ambiguity, and generating meaningful responses even though no specific task is present [45]. To overcome these obstacles, data-driven approaches are often used. In fact, large-scale datasets and machine-learning techniques are used to learn conversational patterns and generate responses [46]. This can also involve external sources of knowledge, such as KGs, to provide informed and relevant answers [47]. As a consequence, when evaluating non-task-oriented dialogue systems, different standards, i.e. regarding coherence, engagement, and naturalness of responses, are applied compared to task-oriented systems [48]. Moreover, due to the subjective nature of non-task-oriented dialog systems, human evaluation plays a crucial role in assessing the quality and effectiveness of these systems [49]. In order to score well and appear human-like, such systems often use mechanisms to control response generation,

like avoiding repetitive or generic responses and ensuring relevance and appropriateness [50]. In the absence of a clear dialogue process, careful consideration of ethical and security issues such as bias, objectionable content, and privacy issues is required [51]. Examples of non-task-oriented dialogue systems are Cleverbot[1], XiaoIce[2] and ChatGPT. The first one is a well-known open-domain chatbot developed by Rollo Carpenter [52]. In order to learn from past user interactions and generate responses for a wide range of conversational inputs, Cleverbot uses an artificial intelligence algorithm. Microsoft created a compassionate social chatbot called XiaoIce which can have informal conversations with users, display emotions, and even produce personalized content, like poetry or stories, based on user inputs [44]. Finally, ChatGPT is a powerful language model created by OpenAI [4]. Although it can be used for a wide range of applications, including task-oriented dialogue systems, it can also engage users in open-domain conversations, generating coherent and contextually relevant responses.

2.1.3. Frameworks for conversational agents

There are numerous frameworks that help developers create a dialogue system or conversational agent (CA). In this section, Google Dialogflow[3] and the Rasa[4] framework are exclusively discussed. In brief, Dialogflow is a closed-source NLU platform with a functional application programming interface (API) and a graphical web interface [53]. The latter, Rasa, is an open-source machine-learning framework for developing text- and speech-based agents or chatbots.

Google's Dialogflow framework provides all the tools necessary to develop a fully functional CA. Its primary and most important features are firstly the extraction of entities and secondly the recognition of the intent from users' utterances. To enable a meaningful conversation flow, Dialogflow works with contexts, where the dialogue data is temporarily stored. This means that the conversation flow is either context- or rule-based. The entities, intents, and contexts can be created not only through Dialogflow's fully functioning graphical user interface but also through an API. If the developer of the CA wants to integrate more logic into the agent, the fulfillment of the response of an intent can also be done via an external API or a webhook service. In general, fulfillment means determining and constructing the response. To facilitate the connection to the user, the integration of the agent with various applications, such as Facebook Messenger [5], Twitter [6], Skype [7] and Telegram [8], is automatically provided by Dialogflow. Moreover, Dialogflow supports text- and voice-based dialogue in multiple languages. Additionally, another benefit of using Dialogflow is that the framework

[1]Cleverbot: https://www.cleverbot.com/
[2]Xiaoice: https://www.xiaoice.com/
[3]Google Dialogflow: https://cloud.google.com/dialogflow
[4]Rasa: https://rasa.com/
[5]Facebook Messenger: https://www.messenger.com/
[6]Twitter: https://twitter.com/
[7]Skype: https://www.skype.com/
[8]Telegram: https://telegram.org/

has well-trained models. However, these models have the disadvantage that they cannot be modified. This inherently means that developers also cannot use their own customized models. Furthermore, as Dialogflow runs in Google Cloud [9], the developer does not have to host it, but communication with Dialogflow during development only takes place via Google Cloud. Consequently, Dialogflow cannot be run on-premise.

As previously mentioned, Rasa is an open-source, machine-learning framework to create chatbots or CAs. It consists of two primary components, the NLU and a component called Rasa Core. Similar to Dialogflow, Rasa's NLU is responsible for extracting the entities and recognizing the intent from the user utterance. The additional Rasa Core is responsible for the fulfillment, whereas Dialogflow does not have such a component. Regardless, both Rasa components are necessary to develop a chatbot. Given the open-source software development kit offered by Rasa, developers can implement their own logic and use their own models. Nevertheless, Rasa also provides state-of-the-art models like the language model BERT, so developers do not necessarily have to use their own model. Moreover, this means that Rasa can be run locally, i.e. on-premise, which can be advantageous during development. In fact, by allowing the addition of custom components, Rasa also supports multiple languages. This framework also integrates easily with applications like Slack and Facebook, providing a quick way to connect with users. Regardless, the main disadvantage of Rasa is that the developer of the CA must have a profound understanding of Python and chatbot development. Additionally, many packages need to be installed as it is mandatory to have the NLU and Rasa Core in order to develop a functional CA.

Based on the above comparison of both agents, we have decided to use Dialogflow as the framework for the agent of this thesis. The most important reasons for our decision were that Dialogflow is easy to set up and superior to Rasa due to the many state-of-the-art models and mature dialogue control concepts already provided by Google. In addition, the prototype of this bachelor's thesis can potentially be integrated in the Alpha-KI[10] project. In this project, smartwatches are used as digital health assistants, whereas other Dialogflow agents have already been deployed. Consequently, using Dialogflow makes it easier to deploy this thesis' developed news agent onto the smartwatch as well as merge it with the other existing agents.

2.2. Knowledge graphs

In the following section, an overview of knowledge graphs (KGs) will be presented and various types of graph databases will be discussed. This will provide sufficient information needed to implement knowledge graph-based CAs focused on news search.

[9] Google Cloud: https://cloud.google.com/
[10] Alpha-KI: https://wwwmatthes.in.tum.de/pages/uysghltybqze/AI-Based-Digital-Health-Assistant-ALPHA-KI

2.2.1. Definition

Given the ubiquitous importance of big data and the increasing complexity of information, KGs have proven to be a useful tool to purposefully organize, process and integrate data in a controlled manner [54, 55]. According to numerous papers, KGs facilitate a wide range of applications, from question-answering and recommender systems to semantic search engines and sentiment analysis [56, 57].

Although KGs are growing in popularity, a clear definition does not exist yet [5]. However, numerous attempts to define KGs exist. According to Ehrlinger and Wöß [54], they can be defined as a structured representation of interconnected entities and their relationships. Moreover, they ease information retrieval, question answering, and decision-making [54]. However, many studies use a generalized, broader definition [5] by Hogan [58]. They define KGs as *"a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities"* [58]. The KGs presented and employed in this thesis adhere to both definition attempts proposed by Hogan and Ehrlinger.

A KG typically consists of the components relationships, nodes, properties, and labels [55]. The KG's relationships and nodes are stored as semantic data triples, as seen in Figure 2.2.

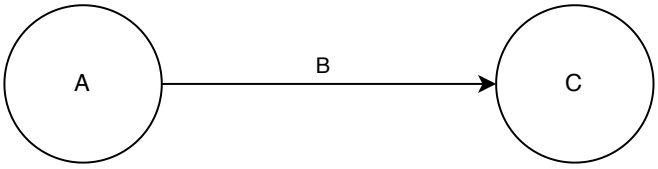


Figure 2.2.: A triple in a directed labeled graph

Nodes A node represents an entity or a concept, such as a person, a place, or an organization. It can have a specific number of properties and can be given different labels. Nodes are connected to other nodes in the graph through relationships.

Relationships A relationship represents a connection between two nodes. They are used to capture the semantic meaning of the connections between nodes in the form of "is parent of", "is located in", or "is a type of" links, for example. A relationship is a directed edge as it has a start node and an end node that are semantically related.

Properties A property in the context of KGs represents an attribute or characteristic of a node or a relationship. Properties are used to store additional information about nodes or relationships, such as their name or a description. They can be seen as a key-value pair

maintained alongside relationships or nodes. One or more properties may be present in a node or relationship.

Labels A KG's label is a descriptive text that is assigned to a node or a relationship. Labels are used to group nodes and relationships that share common attributes or characteristics, making it easier to query and analyze the graph. For example, all "Person" nodes which are "Doctor" get assigned the label "Doctor".

Figure 2.3 depicts an example of a KG wherein the example is taken from [59]. The graph consists of "Movie" nodes colored in orange and "Person" nodes colored in blue. A "Movie" node has the properties "id", "released", "tagline" and "title". As an example, the node of the movie Apollo 13, has the id "314", was released in 1995, has the tagline "Huston, we have a problem" and the title "Apollo 13". A "Person" node has the properties "id", "born" and "name". As an example, the node of the person Tom Hanks has the id "118", was born in 1956, and has the name "Tom Hanks". The "Person" nodes can be connected by the relationship types "DIRECTED" and "ACTED_IN" with "Movie" nodes. The "DIRECTED" relationship has an "id" as its only property. The "ACTED_IN" relationship has the properties "id" and "role". For example, the "ACTED_IN" relationship between the nodes Tom Hanks and Apollo 13 has the id "83" and the role "Jim Lovell". As one can see in the graph, a movie can be directed by more than one person. For example, the film Cloud Atlas was directed by Lana Wachowski and Tom Tykwer. In addition, one person can also be the director of multiple films, e.g. "DIRECTED", as Ron Howard directed both Apollo 13 and The Da Vinci Code. One person can also act in more than one movie, and it is possible for two or more persons to have acted in a movie. As such, Tom Hanks has acted in the films Apollo 13, The Da Vinci Code, Cloud Atlas, and That Thing You Do, and the film Apollo 13 has not only the actor Tom Hanks but also Kevin Bacon in it. It is also possible that a person was active as both a director as well as an actor, such as Tom Hanks, who directed the film "What Thing You Do", but also participated as an actor. However, this cannot be recognized in Figure 2.3. But, to simplify queries, one could now give the "Person" nodes Kevin Bacon and Tom Hanks, the label Actor, and the persons Ron Howard, Lana Wachowski, Tom Tykwer, and Tom Hanks, the label Director. This also demonstrates that a node can have multiple labels. In this case, the node named Tom Hanks would have the labels Actor, Director, and Person.

In order to construct a KG, one must manage to connect structured data with semi-structured and unstructured data. Then, one must extract entities from this data and determine relationships between them. To accomplish this, there are several possibilities, whereas [60] identifies three main categories. They include manual creation, semi-automatic creation, and fully automatic creation. The first is the case when only domain experts can create and maintain the graph. It has the highest accuracy, but also the highest cost, because it takes the longest time and the most work. Secondly, semi-automatic creation combines human expertise with automated tools from domains such as NLP and machine learning. Finally, if one wants to construct a KG fully automatically, only algorithms are used to extract

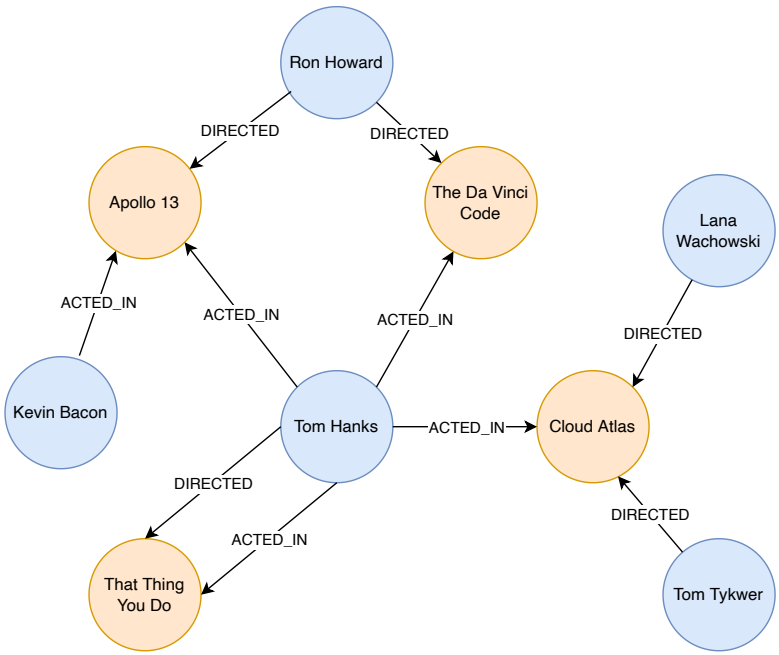


Figure 2.3.: Example themed knowledge graph, based on [59]

information from the different data sources, e.g. texts, images, and structured data, and to connect them together.

KGs are used in a wide variety of systems [61]. A few examples include question-answering systems, recommender systems, semantic search systems, sentiment analysis, entity linking, and conversational interfaces. In addition, there are various domain-specific use cases from different areas, such as finance, education, or medicine. The most famous specific examples are the Google Knowledge Graph, which enhances search results by providing relevant information and relationships, based on a vast collection of structured data [62]. Furthermore, to extract structured information from Wikipedia and to make this data available on the web, a crowd-sourced KG called DBpedia exists [63]. Moreover, YAGO, which covers a vast knowledge base with general knowledge about people, cities, countries, movies, and organizations [64], uses data from Wikipedia and WordNet. Created by the Wikimedia Foundation in 2012, Wikidata is a collaboratively edited, multilingual knowledge base to provide structured data for all Wikimedia projects, including Wikipedia, Wikimedia Commons, Wikivoyage, and others [65].

KGs are a widely used technology for countless use cases. Nevertheless, they still have challenges and problems [61]. Firstly, the scalability of KGs is problematic because they are continuously growing. This leads to dealing with large and ever-expanding data sets. Additionally, a high level of data quality must also be ensured as the information stored in the graph has to be accurate, consistent, and complete in order to provide a correct response

to queries. Moreover, as KGs increase in complexity, it becomes difficult to adapt the graph to data model changes while simultaneously maintaining the correctness and consistency of the graph.

2.2.2. Types of graph databases

In this subsection, we will discuss two major types of graph databases used in KG representation namely the so-called property graphs and the Resource Description Framework (RDF). These two graph models have distinct characteristics and query languages, catering to different requirements and use cases. Firstly property graphs will be reviewed, highlighting their flexible data model and query languages. Afterward, we will further analyze the RDF, examining its standardized data model, Uniform Resource Identifier (URI) usage, query language, and ontology support. The goal is to profoundly understand the differences between these models as this will help in selecting the appropriate approach for implementing knowledge graph-driven CAs in news exploration tasks.

Property graphs First of all, a property graph is a graph model which is made up of nodes, relationships, properties, and labels [66]. Both nodes and relationships each have a specific name and can store properties, which are represented by a key-value pair [67]. Labels are used to categorize nodes into groups. As previously mentioned, edges representing relationships in property graphs are directed and have a start and end node. Therefore, property graphs are directed graphs. Furthermore, property graphs have a special feature as relationships can also have properties associated with them. Thereby, storing additional metadata and semantics for the relationships of the nodes is possible [67]. Also, property graphs are characterized by a flexible data model because new properties can be added or removed easily by directly storing key-value pairs as properties at the nodes and relationships [68]. Given property graphs' schema-less or schema-optional nature, no strict schema requirements do exist. In certain use cases, this can be advantageous because one is not bound to the data schema, but can adapt to the new requirements quickly in fast-changing environments, like e-commerce stores or social networks [56]. A standardized query language to query a graph database that uses a property graph model does not exist. Exemplary query languages are Cypher for Neo4j [11], Gremlin for Apache TinkerPop and AQL for ArangoDB [69].

Figure 2.4 represents a property graph example. It is made up of three nodes and two relationships. The first node is labeled with the label Person and it has the property with the key name and value "Alice". The second node has the label Movie and has two properties. The first has the key name with the value "Inception" and the second has the key year and the value "2010". The third node is labeled as a Genre and has one property name with the value "Sci-Fi". The node Person is connected with the node Movie via a "WATCHED" relationship, which has a property called rating. The Movie node has a second relationship with the Genre node.

[11] Neo4j: https://neo4j.com/

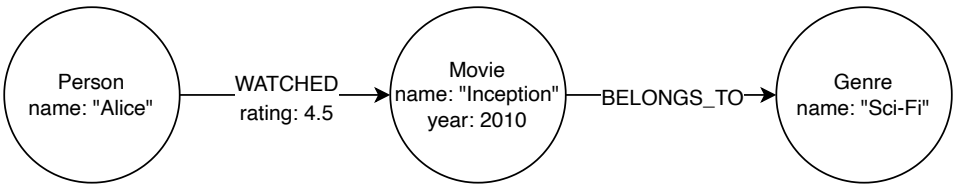


Figure 2.4.: Property graph example

Resource Description Framework The RDF was developed by the World Wide Web Consortium as a model for metadata [70]. They explain that the reason for its development is to make the World Wide Web machine-readable [70]. The basic data model consists of resources, properties, and statements [70]. Resources can, for example, be a web page or a book. A resource is always identified by a unique URI. For a website, it can be the URL and for a book, for example, the ISBN number [70]. Properties are specific characteristics, attributes, or relations to describe a resource. RDF statements consist of a specific resource, a named property and the value of the property [70]. These three parts of the statement are also called subject, predicate, and object as depicted in Figure 2.5.

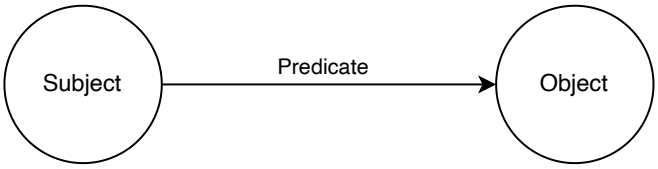


Figure 2.5.: Basic RDF triple

The statement's object, which was described earlier as a property value, can either be another resource, which is then specified by a URI, or simply a basic string [70]. If the object is just a string, it is called a "literal". An RDF database is viewed as a collection of these RDF graphs [66]. As RDF databases use a standardized data model, which stores data as triplets of subject, predicate, and object, data compatibility and interchange are facilitated [71]. Additionally, this data exchange is further simplified by the use of URIs, since using URIs, resources and properties can be uniquely identified globally [57]. Also with the support of ontologies, which is a formal description of domain knowledge, the data can be transferred easily into other systems [72]. Compared to Property Graphs there is the standardized query language SPARQL [73] for RDF, which supports complex graph patterns and combined queries.

Building upon the previous example from Figure 2.4, all RDF statements are listed in Table 2.1. The subjects of the example include "Alice", "Inception", "SciFi", and the new addition, "AliceInceptionRating". Each of these subjects is connected to an object through the predicate "is type of". In this instance, "Alice" is linked to the object "Person", "Inception" to "Movie", "SciFi" to "Genre", and "AliceInceptionRating" to "Rating". All these objects are of the type Resource.

Table 2.1.: RDF statements

Subject	Predicate	Object	Object type
Alice	is type of	Person	Resource
Inception	is type of	Movie	Resource
Inception	was released in	2010	Literal
SciFi	is type of	Genre	Resource
Inception	belongs to	SciFi	Resource
Alice	watched	Inception	Resource
Alice	rated	AliceInceptionRating	Resource
AliceInceptionRating	is type of	Rating	Resource
AliceInceptionRating	scored	4.5	Literal
AliceInceptionRating	movie	Inception	Resource

Moreover, the subject "Inception" is connected to the object "2010" of type Literal through the predicate "was released in". "Inception" is also linked to the object "SciFi" of object type Resource via the predicate "belongs to". "Alice" has "watched" the movie "Inception". Since RDF cannot store attributes with predicates, representing that "Alice" rated the movie "Inception" with a score of "4.5" is more complex than in property graphs. Consequently, "Alice" is connected to "AliceInceptionRating" rather than directly to "Inception" through the predicate "rated". "AliceInceptionRating" is then linked to the Literal "4.5" via the predicate "scored". Finally, "AliceInceptionRating" is connected to "Inception" through the predicate "movie".

To facilitate visualization, the example is also depicted as a graph in Figure 2.6. In this representation, the abbreviation "ex:" stands for the example namespace "http://example.com/", while "rdf:" refers to the namespace "http://www.w3.org/1999/02/22-rdf-syntax-ns#".

Choice for this thesis prototype (Neo4j) Even though both the RDF and the property graphs, have their fair advantages, we have decided to use Neo4j, a property graph model [56], for the development of this thesis' prototype. In short, Neo4j is a major graph database management system. It is intended to store and manage data in the form of a graph. As query language, Neo4j uses Cypher, which is a declarative and pattern-based language explicitly designed for querying graph data [56]. Given Neo4j's efficient performance for graph operations provided by its native graph storage and processing capabilities, it is a popular choice for multiple applications [56].

Moreover, Neo4j offers a more flexible schema, compared to an RDF-based graph database, which can be favorable when working with news data that may have different structures or change over time. Further solidifying the choice for using Neo4j instead of an RDF-based graph database is its native query language Cypher, which is intuitive and easy to learn in contrast to SPARQL. Also, the development of the prototype is facilitated by libraries such as

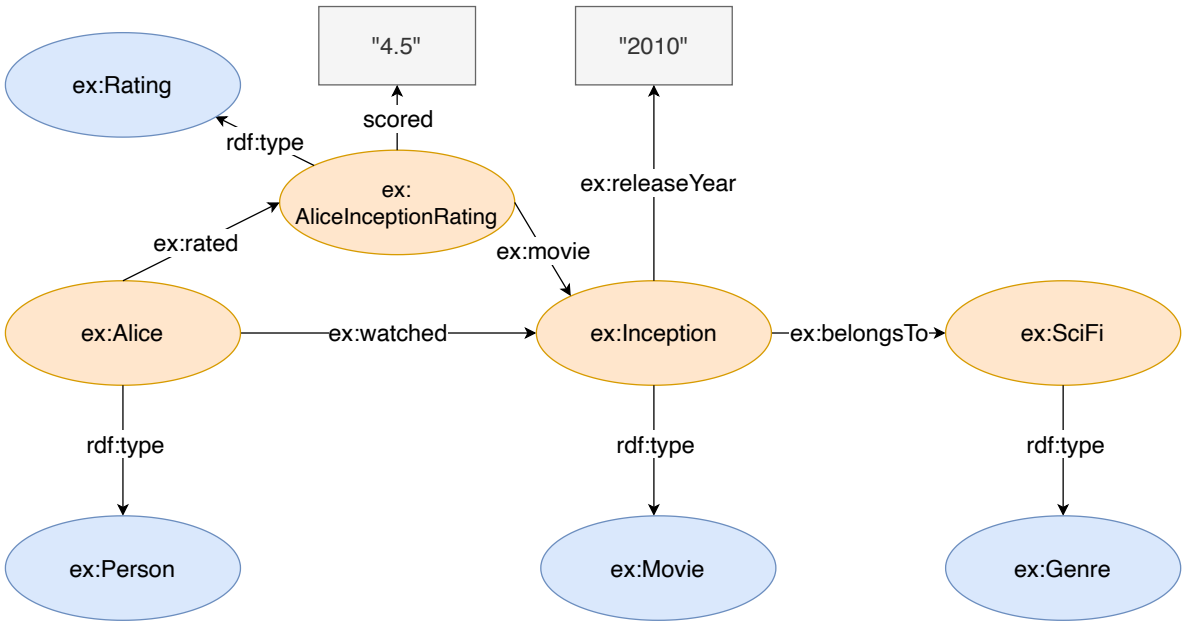


Figure 2.6.: RDF graph example

APOC, which includes many functions such as data loading. Another crucial advantage of Neo4j is that, through the data model structure of the property graph, we hope to achieve a better and simpler representation of the news data. Also, given the vast amount of tutorials and cloud services like the Neo4j AuraDB, learning Cypher and using Neo4j to develop this thesis' prototype is more feasible.

2.3. Combination of conversational agents and knowledge graphs

In this section, we explore the many advantages of integrating CAs with KGs to build more robust and flexible dialogue systems. We can improve information retrieval, natural language comprehension, and reasoning skills by combining these two technologies, all while delivering a smooth and rich user experience.

There are many reasons for the increasing popularity of KG in the field of NLP including their usage as a knowledge base CAs [5]. KGs enable CAs to retrieve information better because their integration can provide more accurate and contextually relevant information to the user, which increases the overall effectiveness of the Dialogue System [74]. By combining the contextual information provided by the KG with users' preferences and their interaction history, the CA can give better-personalized and more tailored recommendations. As a result, the user experience becomes more engaging and satisfying [75]. Moreover, KGs are necessary for having more context-aware conversations as entities and relationships used in the conversation can be saved, which helps the agent link the user input to the broader context of the ongoing conversation [76]. As such, by providing the CA with semantic information and the

relationships between entities, KG improve the natural language capabilities. This means that the additional context supplied by the KG allows the CA to better understand the user intent even in ambiguous or complex situations. In turn, this results in more accurate and relevant answers improving the quality of the conversation [77]. Furthermore, another benefit of using KGs is that they can be easily expanded and enlarged in order to integrate more information and knowledge into the CA [55]. In addition, one can incorporate a lot of information from many domains into a KG. Consequently, one can have multi-domain conversations without having to use a new agent with another knowledge base [78]. Given the access to a rich and interconnected knowledge base provided by the KG, the CA can more easily retrieve information and link it to the conversation's stored context and thereby complete complex tasks more conveniently [79]. Additionally, the structured knowledge base enables the CA to improve its reasoning skills. The relationships and patterns within the KG allow the CA to connect different pieces of information to understand the underlying logic and dependencies. Given this more profound insight, the agent can draw more complex conclusions. Therefore, this ability to reason, in turn, allows the agent to deliver more discerning answers and also to take on decision-making tasks [80].

In summary, by providing more accurate and context-relevant information, more personal recommendations and seamless multi-domain conversations, the user experience increases significantly through the combination of KGs with CAs [81]. CAs can move beyond simple question-answering systems to become more intelligent, contextual, and inferential by utilizing the power of KG, thereby providing users with a more engaging and valuable conversational experience [80].

3. Related Work

In this chapter, relevant publications closely related to the topic of this thesis are presented. Initially, publications concerning conversational agents (CAs) and knowledge graphs (KGs) are introduced, and their synergies are highlighted. Subsequently, existing studies in the field of voice-based news search are discussed. Finally, the distinctions between this thesis and others are elucidated, as well as the unique aspects of this research.

3.1. Conversational agents and knowledge graphs

In recent years, KGs have proven to be a powerful representation of data for a wide range of tasks in the field of NLP [5]. The mentioned publication also highlights that KGs are frequently employed in the development of CAs. This is attributed to their common connection with a knowledge base, which provides the system with organized information. Building upon this, CAs utilize KGs to generate more informative and contextually relevant responses. Previous research efforts that combined KGs with CAs have demonstrated improvements in utterance understanding, dialogue management, and response generation [5].

In addressing the challenge of understanding a user's intent based solely on a brief utterance, Zhou et. al [82] employed word-oriented and concept-oriented KGs to enrich conversational data with more context. The authors developed a novel dialogue system architecture that aligns semantic representations of words and entities to accurately capture individual preferences. In extensive experiments, the authors have demonstrated that their knowledge-based system exhibits improved performance in recommendation and conversation tasks.

Another study, conducted by Xu et al. [83], introduces a dialogue system for automatic medical diagnosis. The dialogue management of this system relies on a medical KG for topic transitions, which enables learning to query symptoms that ultimately lead to a final diagnosis. To evaluate the system's performance, the authors consider both automatic and human assessments.

In the generation of dialogue responses, KGs have taken a significant role in numerous studies, as the following three investigations illustrate. Initially, Chen et al. [84] propose an innovative framework that connects recommendation systems and dialogue generation systems. They show how the recommendation system, using KGs and information from dialogue history, can generate high-quality responses. In another study, Chaudhuri et al. [85]

recognize the value of KGs for response generation. They demonstrate that their approach, which utilizes KGs, generates superior responses for both task-oriented and non-task-oriented dialogues. A third notable study introduces the ConceptFlow model [86]. This model employs KGs to model the course of conversations. It views the dialogue progression as a journey through the KG, navigating along the relations of common sense to related concepts. This approach results in semantically richer and thematically more relevant dialogue responses. These studies show how the structured information in KGs can serve as a grounding mechanism for the responses generated by large language models based on the transformer architecture.

Furthermore, the literature includes a study that employs a CA to interact with a semantically-rich KG [87]. This innovative system facilitates access to Wikipedia articles, thereby allowing the user to pose questions on a wide array of topics and navigate among them. The study bears a resemblance to our work as it also emphasizes accessing information from semantically-rich KGs. However, the distinction lies in the fact that the author does not specifically address the realm of news search and exploration. Despite this difference, the study provides valuable insights and approaches that inspired our work.

3.2. Voice-based news search

Prior to our work, studies on news search and exploration have already been conducted. A significant user study was carried out by Newman in 2018, which investigated user behavior when interacting with smart speakers for news searches [88]. Key areas of his study encompass the prevalence of smart speakers in the United States, the United Kingdom, and Germany, as well as the general development of this technology. Newman also examined the specific dissemination and evolution of news consumption via smart speakers and analyzed how the behavior of news publishers and technology platforms developed concerning news consumption. In the process, he identified several issues related to smart speakers and news consumption. By surveying a large number of users and companies, Newman was also able to capture the corresponding user preferences and trends. These insights are of particular interest to our work, as we can incorporate the many identified problems and user desires into the design of our prototype system.

Following the study conducted by Newman, we would like to highlight another interesting investigation: The study "Would you like to hear the News? Investigating Voice-Based Suggestions for Conversational News Recommendation" by Harshita Sahijwani [89]. This research explores how to best recommend news to users in order to make it interesting for them. To achieve this, voice-based personal assistants are examined. The focus of this study is not on which news items are suggested to the user, but rather on how the news items are presented to the user. Possible approaches include generic, trending news, news briefings, and entity-based recommendations. Through a large number of conducted conversations and their meticulous evaluation, it was demonstrated how well the respective approaches

work. These insights are, in turn, important for the design of our system, as they can help us develop an effective method for presenting news recommendations to our users.

In addition to the studies discussed so far, we would like to highlight another relevant investigation in the field of news search, conducted by Koichiro and Tatsuya Kawahara [90]. In this study, a news navigation system was developed, focusing primarily on its question-and-answer functionality. A remarkable feature of this system is the attempt to never return an error message to the user, but instead always suggest interesting information or news items. This approach aims to enable natural interaction, as not only factual questions should be considered.

The work of Koichiro and Tatsuya Kawahara [90] thus expands on the previous approaches by Newman and Sahijwani with an innovative methodology for designing news search systems. The combination of insights from all three studies can support us in developing a comprehensive and user-friendly prototype system that effectively meets the needs of users in the field of news search and exploration.

3.3. Novelty of our approach

Although numerous publications exist that focus on the integration of CAs and KGs, with some even conducted in the context of information retrieval, to the best of our knowledge, as of April 2023, there are no studies that combine the concept with news exploration. While other news exploration systems and in general voice-based news search have been investigated, no research has yet examined the connection between these two concepts. Based on our analysis of the related literature on conversational search systems, we are the first to implement a system that constructs a KG from German news articles and integrates it with a voice-based CA for news exploration. This underscores the innovative strength and contribution of our work in the field of CAs, KGs, and news exploration.

4. Method

In this chapter, we outline the methodology employed to address the research objectives of this thesis. The process is divided into six main steps, which are described in detail below and as illustrated in Figure 4.1.

(1) Conduct literature analysis about state-of-the-art in voice-based news search

In the first step, we conducted a brief literature analysis to identify the state-of-the-art in voice-based news search systems. We compared popular voice assistants, including Siri, Amazon Alexa, and Google Assistant, in the area of news search to understand their capabilities and limitations. Additionally, we reviewed a few relevant studies such as the Newman study [88] and the work by Sahijwani et al. [89] to gain insights into the current landscape, challenges, and potential opportunities for improvement in this area.

(2) Construct knowledge graph (KG) based on German news data

In the second step, we constructed a KG using Neo4j, as explained in 2.2.2. We retrieved news data from the Tagesschau API[1], which is one of the most reputable news providers in Germany. To process and represent the news articles in the KG, we employed the Wikifier[2] to extract entities from the article text. After the extraction, we classified these entities into classes such as Person, Company, and others. This structured representation of news data in the form of a KG allows for efficient querying and retrieval of relevant information, which is crucial for our conversational agent's (CA) functionality.

(3) Build prototype of CA and integrate with KG

In the third step, we built a prototype CA using Google Dialogflow and integrated it with the previously constructed KG. We modeled potential user queries and designed intents and entities within the Dialogflow agent to effectively handle user inputs.

To connect the Dialogflow agent with the Neo4j KG, we developed a webhook service. This application serves as an intermediary between the agent and the graph, enabling the retrieval and processing of relevant information from the KG in response to user queries.

[1]Tagesschau API: https://tagesschau.api.bund.dev/
[2]Wikifier: https://wikifier.org/

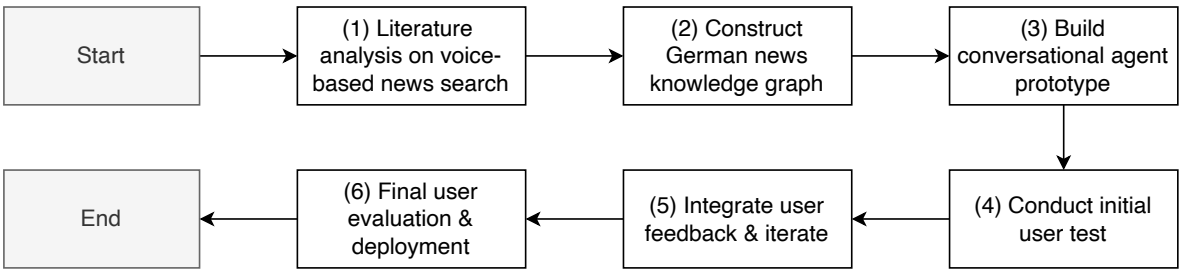


Figure 4.1.: Steps of the research method

Furthermore, we created a simple frontend using Python Flask, which includes a button for recording the user's voice input. This frontend is designed to facilitate user interaction with the CA in a more intuitive and user-friendly manner, making it suitable for the upcoming user tests. The frontend handles the recording, sending, and playback of user input and agent responses.

(4) Conduct first user test to evaluate basic capability

In the fourth step, we conducted the first user test to evaluate the basic capability of the developed CA. We invited a small sample of users to interact with the prototype, simulating a realistic usage scenario. The primary goal of this initial testing phase was to gather feedback on the system's usability and overall user experience.

During the test, we encouraged users to provide both positive and negative feedback, focusing on areas such as the agent's voice, advantages and disadvantages of the agent for news search, the overall user experience, the quality of information provided, and any suggestions for improvement. We also asked users about any additional functionalities they would like to see in the system.

This feedback was crucial for identifying potential issues, areas for improvement, and any unanticipated user needs or expectations that could be addressed in the next iteration of the prototype.

(5) Integrate user feedback in second iteration to improve capabilities

In the fifth step, we incorporated the feedback received from the initial user testing to enhance the capabilities of the CA. This iterative approach allowed us to refine the prototype, addressing any identified issues and integrating the desired functionalities mentioned by the users.

Some of the improvements made during this phase included:

- Improved the agent's voice to make it more natural and pleasant.

- Enhanced visual feedback and recording capabilities in the frontend for a better user experience.
- Added more control over article selection, such as selecting articles by keywords.
- Enabled users to have more control over reading articles, such as repeating or skipping an article.
- Implemented better suggestions to users, providing more relevant and tailored recommendations.

The result of this phase was an improved version of the prototype, which was then prepared for the final user evaluation. This iterative development process ensured that the system was better aligned with user needs and expectations, thereby increasing the likelihood of a successful final evaluation.

(6) Final user evaluation and deployment

In the sixth and final step, we conducted a more extensive user evaluation of the improved prototype, involving a larger number of participants. The goal of this evaluation was to assess the overall performance and usability of the CA, as well as to identify any remaining issues or potential areas for future improvement.

During this evaluation, users interacted with the CA, providing feedback on various aspects, such as the quality of the voice, the relevance of news recommendations, and the usability of the system. This comprehensive evaluation allowed us to gather valuable insights into the strengths and weaknesses of the prototype, which can be used to guide further development and refinement of the system.

5. Results

In this chapter, we present the results of our research, addressing each of the research questions posed at the beginning of this thesis. The chapter is organized into five sections, one for each research question. We discuss the findings for each research question, which collectively contribute to a comprehensive understanding of voice-based news search and exploration using a knowledge graph (KG). It should be noted that a selection of these results has already been published in a paper [91]. This includes the architecture of the prototype, which is explained in subsection 5.4.1, as well as the graph data model that is described in subsection 5.3.1. Furthermore, the literature review on the Newman study and the experimental analysis of commercial voice assistants which are presented in section 5.1.

5.1. Research question 1

To address the first research question *"What is the current state-of-the-art in voice-based news search and exploration?"*, we further analyze the study conducted by Newman [88], introduced in Chapter 3. Additionally, we conducted an experimental analysis comparing the capabilities of Siri, Amazon Alexa, and Google Assistant in the area of news search. Our findings are presented below.

5.1.1. Literature review: Newman study

Newman [88] found that most users consume news on a daily basis through brief updates via command and control. Many users felt overwhelmed by the technology and the volume of news, and they appreciated that voice-based interfaces provided precise information in an easily understandable format. Moreover, people had the impression that by navigating the news with their voices, they had more control over the device. Some users found voice interaction to be more natural and intuitive.

In his study, Newman noted that consumers did listen to the news in the morning using voice-based agents, but they wished that the news would be shorter, lasting only up to a minute, and more up-to-date. However, most users complained about insufficient control over the search process and found the news briefings to be too long. The current news was often too detailed and recorded with poor sound quality. Furthermore, the news briefings were frequently interrupted by intrusive advertisements or jingles, which negatively impacted user-friendliness and overall satisfaction.

Another obstacle uncovered by Newman was the frequent repetition of news updates when using different news providers. Users expressed a desire for more detailed information on specific topics and the ability to select or skip individual stories when not interested. Most users consumed news only from well-known and leading news sources. Many users wished to compile their own short news briefings with news articles that interested them and listen to similar articles after reading one. Newman found that many users wanted to ask more detailed and complex questions about an event or person featured in an article. However, 10% to 15% of users were unsure of how to access or use the news functionality and would appreciate some assistance. Some users had privacy concerns and preferred to press a button to start recording instead of being listened to all the time. In summary, Newman's study identified that topic-specific searches, personalized news content, and increased control over navigation could significantly improve the overall user experience.

5.1.2. Experimental analysis of commercial voice assistants

Existing voice assistants, such as Amazon Alexa, Google Assistant, and Siri from Apple, offer news search as part of their functionalities. With roughly 80 million monthly active users each, these three voice assistants are the most popular and market leaders [3]. To get an overview of this area we systematically tested all three assistants using a predefined question catalog in order to compare their capabilities. During this experimental assessment, we discovered that these commercial assistants are especially limited regarding exploratory news search and often lack depth of content.

Based on our systematic comparison of the three mentioned assistants Siri, Alexa, and Google Assistant, which we conducted in December 2022, it became evident that there is substantial room for improvement in voice-based news search, although the status quo offers already many features. To compare these assistants, a list of questions with typical user statements in the domain of news search was defined. These statements pertain to a broad news search where the user receives an overview, news from a specific location, news about a particular person, as well as various control instructions for better navigation in the news output. For testing Siri, an iPhone XR with iOS 16.0 was used, for testing Google Assistant, the iPhone app "Assistant" with version 1.9.64101 was used, and for Amazon Alexa, the Amazon Alexa Echo Dot of the fourth generation was utilized. The key findings indicate that the features and issues of the three assistants are quite similar and align with the results of Newman [88]. Table 5.1 summarizes the experimental results for each examined voice assistant.

Upon the request "Tell me the news", all three agents only return a webpage with web search results for the user query. This means that users need to know precisely which keywords, such as "play," they must use in their request to receive a spoken response. However, even when the correct keyword is used in a query, the systems often play podcasts, some of which are longer than 10 minutes. The fundamental control commands work relatively well, but they are limited to playing the next podcast, repeating, or pausing the current podcast. Siri

and Google Assistant exhibited some difficulties in skipping or repeating a podcast. Moreover, many of the suggested podcasts have similar content, confronting users with much repetitive information. After a user has finished listening to a news podcast, the tested voice assistants do not suggest related news on the same or similar topics.

Table 5.1.: Summary of experimental test results for each voice assistant [91]

Focus	**Search query**	**Siri**	**Alexa**	**Google Assistant**
General	”Tell me the news.”	Displays web search result page for the query.	Plays the ”100-seconds Tagesschau” podcast.	Plays the ”100-seconds Tagesschau” podcast.
	”Play the news.”	Plays the ”100-seconds Tagesschau” podcast.	Plays the ”100-seconds Tagesschau” podcast.	Plays the ”100-seconds Tagesschau” podcast.
Category	”Tell me the news about sports.”	Plays the ”kicker news” podcast (1-2 minutes).	Gives an update about the biggest sports clubs.	Plays the ”100-seconds Sportschau” podcast.
	”Tell me the news about politics.”	Cannot retrieve any news.	Plays the ”100-seconds Tagesschau” podcast.	Plays podcast about Kosovo and Serbia conflict (13 minutes).
	”Tell me the local news.”	Plays the ”Deutschlandfunk” podcast (10 minutes).	Plays the ”100-seconds Tagesschau” podcast.	Cannot retrieve any news.
Topic	”Tell me something about the Ukraine war.”	Plays podcast about the Ukraine war (4 minutes).	Tells one fact about the Ukraine war.	Displays web search result page for the query.
	”Play the news about the world cup.”	Displays web search result page for the query.	Tells some general facts about the world cup (30 seconds).	Displays web search result page for the query.
	”Play the news about Donald Trump.”	Plays podcast that contains the name Donald Trump.	Tells a summary about Donald Trump.	Displays web search result page for the query.
Control	”Next.”	Cannot skip podcast.	Plays next podcast.	Plays next podcast.
	”Again.”	Repeats podcast.	Repeats podcast.	Cannot repeat podcast.
	”Pause.”	Pauses podcast.	Pauses podcast.	Pauses podcast.
	”Play.”	Plays podcast.	Plays podcast.	Plays podcast.

When asked for news from a specific category, a random podcast from the category is played. Often, a podcast with the name of the requested area in the title is simply played. Consequently, users are not informed about current events from the category but may receive a podcast entirely unrelated to the topic. For example, in our experimental analysis, we asked for news about politics and Google Assistant played a 13-minute-long podcast about the Kosovo and Serbia conflict. Although this is a political subject, the podcast is not about

current events but about the general history of the conflict, which is not related to daily news and represents only a small portion of current political news. Another observation is that these systems do not support entity-based news search, meaning that users cannot ask for news on a specific topic, such as Donald Trump or the World Cup, as shown in Table 5.1. In the latter case, the tested voice assistants respond with facts, internet search result pages, or podcasts, rather than retrieving news articles. Since these systems do not possess a knowledge representation of current news topics, they struggle to locate local news and news about specific events. Overall, the experimental results demonstrate that there is significant potential for improvement in voice-based news search. A topic-specific search, personalized news content, and more control over navigation could considerably enhance the overall user experience.

In summary, our analysis shows that current voice-based assistants exhibit considerable limitations regarding news search. Users have difficulties obtaining relevant and personalized news content, and control over navigation is limited. To overcome the identified issues, in the next chapter, we will explore potential approaches to address these challenges and improve the user experience when using voice-based assistants for news search.

5.2. Research question 2

To address Research question 2 *"What are suitable interaction patterns for voice-based German news search and exploration?"*, we will discuss suitable interaction patterns for improving voice-based German news search. The proposals and approaches for interaction patterns are based on the results of our literature analysis and our experimental analysis of voice assistants. The improvements are mainly in three primary areas: topic-specific search, also known as entity-specific search, more personalized content, and enhanced navigation control. After identifying the opportunities that might lead to improvements, we developed a finite state machine taking the previously detected problems into account and improving them. This finite state machine also serves as a proposal for the basis of the prototype developed in this thesis.

The mentioned Finite State Machine is depicted below in Figure 5.1. In brief, on the left-hand side of the figure, the user begins at the "Start". From there, the user has two choices: he either greets the system or asks it for help. When the user greets the system, he is in state 1 Greeting (S1) and when the user asks the system for help he is in state 2 Help (S2). As seen in the figure, at the beginning of state 3 News Exploration (S3), after greeting or asking for help, the user can start exploring news.

Next, in S3, the user has three choices: firstly, the user has the option to get a daily overview, secondly, to ask for news from a certain category such as "politics" or "sports". As a third option, the user can ask for news about a certain entity for example a person or country. If the user chooses the first option, receiving a daily overview, he goes to state 4 Overview Search (S4). The state of the category search is state 5 (S5) and if the user is searching for news about

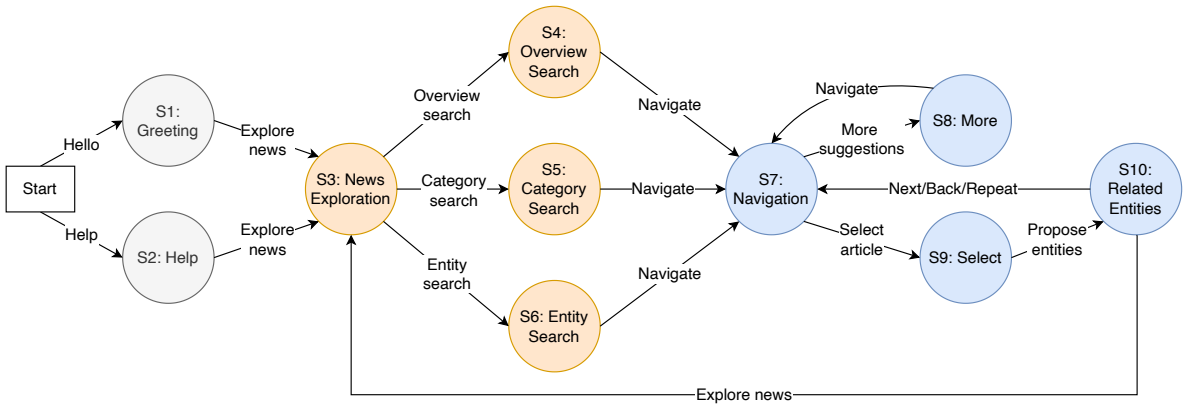


Figure 5.1.: Finite-state machine of conversational agent

a specific entity, the user is in state 6 Entity Search (S6). In state 4, three headlines of current articles from different news areas are suggested to the user. In S5 and S6, three headlines of current articles about the selected category or entity are presented to the user.

After all three news exploration states, the user is forwarded to the Navigation state (S7), as the user can now navigate through the selection. First of all, if none of the three suggested articles is of interest to the user, he can ask for further suggestions. This would put the user in state 8 More (S8). Once in this state, the system will present three additional articles to the user either from the overview articles, from the selected category, or for the selected entity. After the suggestion of the articles the user is back in the Navigation state (S7).

If the user discovers an interesting article immediately or after any number of further suggestions, the user can select an article and is then in state 9 Select (S9). After the system reads the article to the user he ends up in state 10 Related Entities (S10). Here, the system proposes three related entities from the article to the user. In this state, the user can either have the article repeat itself or select the next or previous article from the before-suggested selection, thus landing back in the Navigation state S7. In addition, the user can also search for other news either from one of the proposed entities or from one of the other options. If the user decides to do so, he will return to state 3 News Exploration.

5.2.1. Help

First, we will discuss the proposed help function. This feature contributes to addressing a part of the issues identified in section 5.1. One reason for integrating this function is that some users in Newman's user study [88] were not aware of the news function or did not know how it works. The help function serves to inform the user directly about the system's features, thereby accelerating the user's learning until he understands the system. As an example, the help function provides the user with a brief overview of the system's features and also includes sample voice commands. This allows the user to obtain a satisfactory

response from the system more quickly, reducing user frustration and enhancing the user experience. After the user has received help, he is asked if he requires additional assistance. This process repeats itself until the user is satisfied and feels sufficiently informed.

5.2.2. Overview search

In this subsection, we explain why we have included a daily overview as an interaction pattern in our system and how it solves a part of the previously identified problems from the section 5.1 regarding research question 1. As described in the section before, the important key points regarding the overview feature were that people want to hear a short news update in the morning that lasts less than one minute. The podcasts currently offered by the existing systems do not address this need due to their length and time-consuming nature. A further problem to be solved regarding podcasts is that they have to be produced first, meaning they update listeners about current news with some delay. In addition, listening to several podcasts at a time leads to repetitions and irrelevant or undesired information. Furthermore, podcasts might be interrupted by distracting music or annoying commercials.

The interaction pattern to solve these problems is that the user gets the option to ask the system for a daily overview. The answer of the system is very simple as it gives the user the headlines of three news articles. Since this thesis is about German articles, the user is provided with one article from each of the categories domestic, foreign, and economy, respectively, in order to cover all of the most common areas of news. To prevent an article from appearing twice, as often is the case with podcasts, the system always returns randomly one of the ten most recent articles from the respective category. An advantage of the system is that articles are ready for news consumption as soon as they are published and incorporated into the system's KG, unlike the time-intensive production of podcasts. Given the option to ask for more news suggestions, which results in the user receiving a total of three more articles, the user himself can regulate the length of the news output. Lastly, since the user can also decide individually for himself which of the articles he wants to be read aloud completely, the user is not bound to prefabricated formats. This means that the article search is more personalized, which improves the user experience.

5.2.3. Entity-based search

In the previous section 5.1, it was identified that it is not possible to ask for news about a specific entity because the user would only receive another podcast, which might cover the topic of interest. To address the previously outlined issues in section 5.1 we argue that an entity-based search is a useful strategy. Entity-based search can be applied to retrieve articles about concepts in the objective world. Sahijwani et al. [89] discovered that acceptance was up to 100% higher for entity-based suggestions compared to trending news and was 29% higher for entity-based recommendations compared to a generic news briefing. They highlight that in order to keep the user engaged, it is critical to offer options that relate to specific entities of interest.

The aforementioned findings were incorporated into this thesis' prototype. As such, we have integrated the function to search directly for news related to specific entities, into our system. For our prototype, we use a KG that is ideal for entity-based news search because it supports semantic queries that are commonly used in this type of search. In addition, the graph structure allows entities to be linked directly to articles, making it optimal for entity-based searches. The implementation is similar to the overview search as three of the 15 newest articles are randomly suggested to the user for the requested entity. This ensures that if the user asks for more news, he does not get the same article multiple times. In the case that a user does not search for news directly for an entity but rather requests news from a category, such as economy, domestic or foreign, he again randomly receives three of the latest 15 articles from the respective news category. As a result, it is avoided that a user obtains the same message numerous times if the user asks for more suggestions from one topic of interest. If the user does not specify an entity in the news search but asks the system to present a selection of available entities, the system suggests three categories and three entities to the user. For the categories, three random categories are simply taken. Regarding the entities, they are determined in such a way that three random entities are selected from the top 15 entities that have occurred most frequently in all articles in the last two days prior to the query. This is done to facilitate a start to the conversation in case he decides against a daily overview and does not know exactly what news he is interested in.

We assume for our system that, if a topic arouses the user's interest, he may also read articles about the same person, country, or other entities mentioned in the article. Therefore, after an article has been read out to the user, the system suggests three entities related to the article, using the ability of the KG to map all relevant information about an article with its entities and relationships to other similar articles. The three suggested entities are selected the following way: the entities are related to the read article. Out of these related entities, those entities are selected that appear most frequently in all other articles. Thereby, the chosen entities result from an overlap of personal preferences and top headlines. Furthermore, entity suggestions are made after each article is read, and it is irrelevant whether the user comes from a daily overview, a category-specific search, or an entity-based search. This enables smooth navigation and coherent topic transitions. The pseudo-code for the algorithm, which proposes related entities for any given news article, is described below:

Algorithm 1 Entity suggestion algorithm

Require: news article A, knowledge graph G
$entities \leftarrow \text{get\_linked\_entities}(A, G)$
$sorted\_entities \leftarrow \text{sort\_entities\_by\_freq}(entities, G)$
$E \leftarrow \text{select\_top\_entities}(sorted\_entities, 3)$
return list of three suggested entities E

The following example of how to find a desired news article through an entity-based search clarifies its benefits: a person has heard that a new restaurant will soon open in her village.

Therefore, she wants to find out if there is any local news about this restaurant but she does not remember the details about this opening event. With our proposed system, the person can easily navigate to existing news articles by using related keywords and entities, such as restaurant, opening, the name of the village, or food, without having to know any specifics about the restaurant, like its exact name, the cuisine, or where it is located. The person does not need to read through an entire newspaper or listen to a long podcast in order to identify the desired information. The example demonstrates how intuitive the associative search based on entities can be.

5.2.4. Control

Users want to have more control over their messages to improve their results, which, in turn, increases the user experience. Although the control commands in the comparison test of SIRI, Alexa, and Google Assistant already worked well, they were only viable for playing the next podcast, repeating the current podcast or pausing the podcast, or playing it again. Therefore, in the expected interaction pattern of the prototype, mechanisms were included to allow the user to control their news consumption more precisely.

First of all, the fact that the user himself can choose which articles are suggested to him increases the user's control over his interaction with the system. In addition, from the self-determined article selection, only those articles are read out to the user that he selects himself. Moreover, it is not only easy and user-friendly to select individual articles, but the user can also repeat them, skip them, or read to him the previous article from the suggested selection. Given the ability to request additional article suggestions as often as the user likes, the user has full control over the length of the news output. Finally, the user can also interrupt the voice output at any time and request messages about other topics or an overview by entering a new voice command. As such, the user has full control over exactly which articles he wants to consume.

5.3. Research question 3

In the previous sections, we discussed the limitations of existing voice assistants for news search and the interaction patterns to overcome these limitations. To improve the overall news search experience even more by providing users with a more personalized and efficient way to find news, we propose constructing a news knowledge graph (KG). This KG focuses on nodes and their relationships to facilitate a more intuitive and relevant news search. The KG of this thesis' prototype is maintained in a graph database. For this purpose, we decided to use the graph database management system Neo4j as described in Section 2.2.2. In this section, we will describe the data model of the news KG, explain the construction process using data from the Tagesschau API, and provide a detailed example to demonstrate the effectiveness of the KG in improving news search for a wide range of users.

5.3.1. News graph data model

In this subsection, we will introduce the data model that serves as the foundation of our news KG. The data model is designed to effectively capture the nodes, their properties, and the relationships between them, which are crucial for enabling an advanced news search.

In Figure 5.2, the data model of the KG is illustrated in diagrammatic form. The KG is composed of distinct nodes, such as articles, reports, tags, entities, and entity classes. Each article is associated with multiple properties, including a unique identifier, a creation date, a title, an opening paragraph, and an article text. In addition, articles can be labeled with so-called tags, which each have a unique name as a property. As such, news articles can have several relationships with tags. The relationship between an Article node and a Tag node is called "HAS_TAG". An article is part of one category, which, similar to tags, only has a unique name as a property. The relationship between an article node and category nodes is called "IS_PART_OF". A news article can have multiple linked entities. The articles and entities are linked with the "HAS_ENTITY" relationship. The entity nodes in the graph are the entities that appear in the text of the article. In an article about the American presidential election, entities like Joe Biden and Donald Trump might appear, for example. All entities have a unique identifier, the wikiDataItemId, a name, and a Uniform Resource Locator (URL) as properties. All constructed entities are categorized into entity classes like city, company, or person. Each class node has only a unique name as a property. The relationship between entity and class nodes is called "INSTANCE_OF".

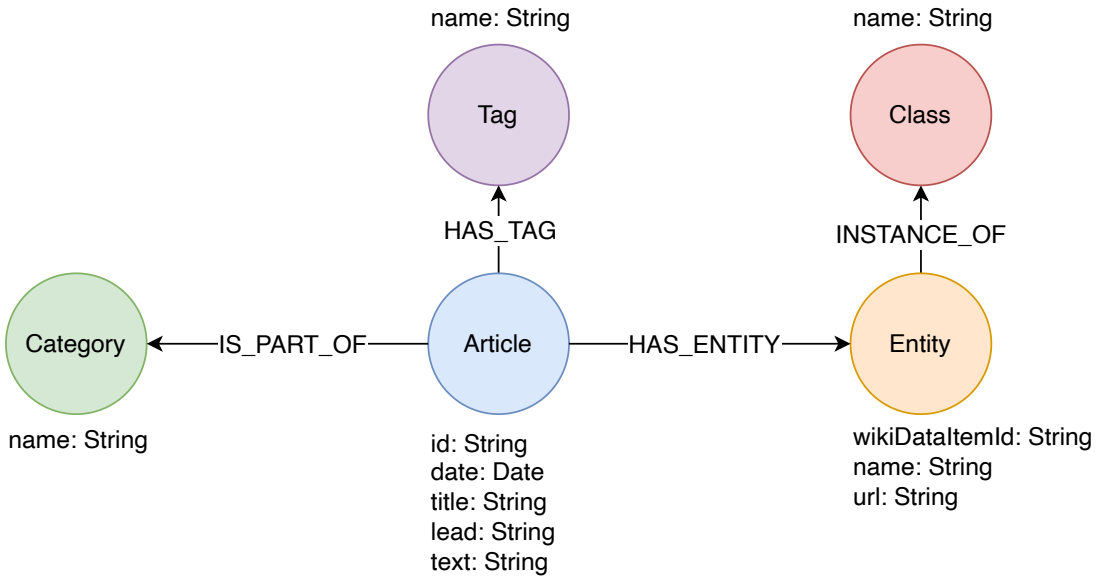


Figure 5.2.: Data model of constructed news knowledge graph [91]

5.3.2. Construction of news graph

In this subsection, we will outline the process of constructing the news KG using data from the Tagesschau API. We will explain how nodes and their relationships are created, and provide insight into the queries used during the construction process.

Tagesschau API The data used to create the nodes is retrieved from the Tagesschau API, which is a service that uses the data of the Tagesschau. The Tagesschau is a television news program of the German public broadcasting association "Arbeitsgemeinschaft der öffentlich-rechtlichen Rundfunkanstalten der Bundesrepublik Deutschland" (ARD). The API provides access to the news content produced by the Tagesschau, which covers an extensive range of categories, including politics, economy, sports, culture, and science. This data is attainable in the structured form of JavaScript Object Notation (JSON) "files" so that the news-related data can be easily retrieved and processed programmatically.

The organized data contains information regarding news articles, e.g. title, publication date, content, images, and associated tags, which can be used to create nodes and relationships in the KG. The Tagesschau API is regularly updated with the latest news articles resulting in the KG always being up to date and delivering relevant search results. Another benefit of using this API is that the news articles can be accessed in German, which is crucial for the German-speaking users of this thesis' prototype's target audience. In addition, the Tagesschau API provides filtering and sorting options that allow developers to retrieve news articles based on specific criteria, such as date range or categories.

APOC The Neo4j APOC Library is used to retrieve data from the Tagesschau API and to create the nodes in the graph with this information. The acronym APOC stands for "Awesome Procedures on Cypher" and is a library of custom functions for Neo4j. This library comprises over 450 functions that can be used to extend the capabilities of Cypher, the query language for Neo4j. As APOC allows data to be retrieved and imported into a Neo4j graph database it has the ability to integrate with external APIs, such as the Tagesschau API. It can handle various data formats, including JSON, XML, and CSV, making it versatile in using different data sources. Other functions of APOC include the refactoring of graphs and various graph algorithms. APOC is an open-source project with an active community of users and contributors who continuously improve and extend its features. Since it can be easily installed as a plugin for Neo4j it is used to create the KG used in this thesis' prototype.

Constraints Before explaining the creation of the nodes and relationships in more detail, we will impose constraints on the graph, which ensure that each node has a unique identifier. Subsection 5.3.1 already explained which identifier each node has. While constructing the nodes, the unique identifiers ensure that no two nodes will have the same identification. The constraints are listed below:

```
CREATE CONSTRAINT IF NOT EXISTS ON (a:Article) ASSERT a.id IS UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS ON (t:Tag) ASSERT t.name is UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS ON (c:Category) ASSERT c.name IS UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS ON (e:Entity) ASSERT e.wikiDataItemId is UNIQUE;
CREATE CONSTRAINT IF NOT EXISTS ON (c:Class) ASSERT c.name is UNIQUE;
```

Creation of article, category, and tag nodes Retrieving the data from the Tagesschau API and thus updating the data in the KG happens automatically every hour. The Cypher query is only two lines long thanks to APOC. The APOC method "apoc.load.json()" is used to retrieve news-related data from the "https://www.tagesschau.de/api2/" endpoint of the Tagesschau API. The "UNWIND" clause is utilized to convert the retrieved JSON data into individual news items.

```
CALL apoc.load.json("https://www.tagesschau.de/api2/") YIELD value
UNWIND value.news AS n
```

After the news items are queried, only the news items that are relevant to our graph are filtered out. In this process, all items that do not have an id and do not belong to a category are identified and left out. In addition, only items of the type "story" are selected, since photo collages and podcasts do not fit our use case. The content is extracted from these news items and the first three text modules, which are either of type text or headline, are chosen to represent the text of the articles. For this purpose, only the first three text modules of the content are selected since they contain the most important information and give an overview of the articles. We do not use the entire text of the article, as it would be too long for voice output.

```
WITH n
WHERE n.externalId IS NOT NULL AND n.ressort IS NOT NULL AND n.type = "story"
UNWIND n.content AS c
WITH c, n
WHERE c.type = "text" OR c.type = "headline"
WITH apoc.text.join(collect(c.value)[..3], " ") AS text, n
```

Next, the article nodes are created and the properties date, title, and lead are set according to the Tagesschau API's news items properties date, title, and topline. Additionally, the property text is set with the value of the previously described composite text. The unique identifier is set by the externalId of the news item. Given the constraints, a new article is only created if no other article with the same externalId already exists.

```
MERGE (article:Article {id: n.externalId}) ON CREATE SET article.date = n.date,
article.title = n.title, article.lead = n.topline, article.text = text
```

In the next step, category nodes are created. For this, it is sufficient to only set the corresponding attribute of the news item. Again, the constraints apply to result in no category node with the same name being created twice. In addition, the "IS_PART_OF" connection between the newly created article node and the category node is established here.

```
MERGE (category:Category {name:n.ressort})
MERGE(article)-[:IS_PART_OF]->(category)
```

Finally, the tag nodes are created. For each entry in the tags list of the news item a new tag node is created and the relationship "TAGGED" from the article to the newly created tag node is established.

```
FOREACH(t IN n.tags | MERGE (tag: Tag {name: t.tag})
MERGE (article)-[:TAGGED]->(tag))
```

Named-entity recognition and entity linking In the constructed KG the entity nodes are all entities that appear in the text of the article, as already briefly mentioned in 5.3.1. Before explaining how exactly they are constructed, the techniques used to create them are explained first. The techniques used are entity recognition also known as named-entity recognition (NER) and entity linking (EL).

NER refers to the process of identifying and classifying named entities, such as people, organizations, or places, in unstructured text, according to [92]. It is identified in [92] that NER, in general, is a subtask of information extraction and is widely used in NLP. In doing so, NER systems typically use machine learning techniques such as decision trees, maximum entropy classifiers, or deep learning models to identify and classify entities [92]. NER can either be rule-based, statistical, or a combination of both [92]. Typical applications of NER include information retrieval, question answering, and text mining [92].

EL is the process of linking textual mentions of named entities to their corresponding entities in a knowledge base [93]. An example of EL is linking "Barack Obama" to the corresponding entity in Wikidata or DBpedia [93]. The main benefit of EL is resolving ambiguities and connecting text data with structured knowledge, according to [93]. It is clarified in [93] that EL usually involves three primary steps, namely candidate generation, candidate ranking, and disambiguation. Firstly, the candidate generation retrieves a list of candidate entities from the knowledge base that could match the textual mention. Secondly, the candidate ranking assigns a score to each candidate entity based on factors such as context similarity, entity popularity, and string similarity. Thirdly, the disambiguation selects the highest-ranked candidate as the correct entity for linking. Applications of EL include question answering, KG creation, and semantic search.

An example is depicted in Figure 5.3. All Wikipedia articles are used as the knowledge base. In the sentence "Apple is headquartered in Cupertino, CA." the entities "Apple" and "Cupertino" were extracted by NER. Regarding the entity Apple the fruit apple, the company Apple Inc., as well as the city Apple in Oklahoma, are available for selection. For the second entity Cupertino, the company Cupertino Electric and the city of Cupertino in California are listed. The EL process results in the decision for the company Apple Inc. based on the Wikipedia article "wikipedia.org/wiki/Apple" and for the city Cupertino based on the Wikipedia article "wikipedia.org/wiki/Cupertino".

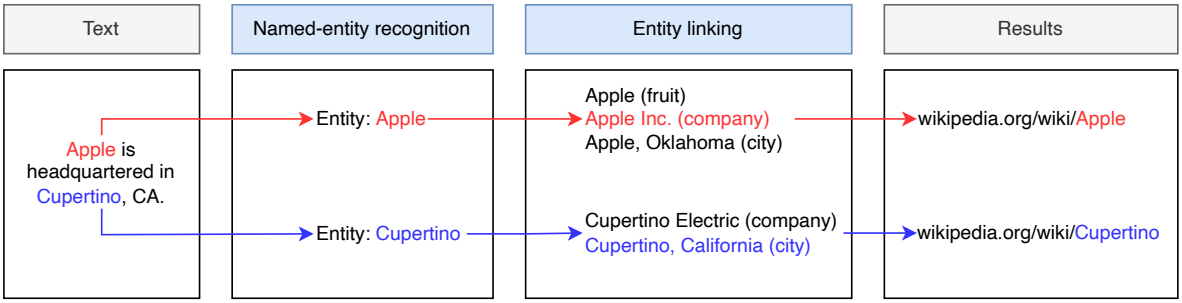


Figure 5.3.: Named-entity recognition and entity linking example

For this process, the so-called Wikifier, which is a NER and EL system developed by Microsoft Research, is used in this thesis prototype. The Wikifier connects entities with the according Wikipedia pages. Consequently, it uses Wikipedia as its knowledge base. Further, Wikifer supports 100 languages and provides an API endpoint that is free to use. An advantage of using the Wikifier is that you can use the wikiDataItemId and the Wikipedia URL to get more information about the entity and also to use services like the Wikidata API.

For example, if you send to this API the sentence from the previous example, you will get the following response in Table 5.2. From the sentence, the entities with the title "Apple Inc." and Wikipedia URL "https://en.wikipedia.org/wiki/Apple_Inc." and the wikiDataItemId "Q312" and another entity with the title "Cupertino, California" and the Wikipedia URL "https://en.wikipedia.org/wiki/Cupertino,_California" and the wikiDataItemId "Q189471" were extracted.

Table 5.2.: Wikifier results

Title	**Wikipedia URL**	**WikiDataItemId**
Apple Inc.	https://en.wikipedia.org/wiki/Apple_Inc.	Q312
Cupertino, California	https://en.wikipedia.org/wiki/Cupertino,_California	Q189471

Entity nodes creation We employ NER and EL to extract entities from the text of articles and create entity nodes within our KG. For this aim, we utilize the Wikifier system. The process is inspired by the approach outlined in [94]. Entity nodes within the KG are useful for enabling features such as entity-based news search or topic suggestions.

Initially, we select all article nodes that have a text and for which entities have not been extracted yet. This is done to avoid repeatedly sending the text of all article nodes through the Wikifier process, thereby improving the performance of the entity recognition and EL processes.

```
MATCH (a:Article)
WHERE a.text IS NOT NULL AND NOT (a)-[:HAS_ENTITY]->(:Entity)
RETURN a
```

In the next step, the Wikifier API request is constructed with the necessary parameters. The language of the Wikifier is set to German, as German articles are stored in the graph. An important parameter is the pageRankSqTreshold, which is set to 0.80 in this case to minimize the misassignment of entities.

```
WITH a, "https://www.wikifier.org/annotate-article?" +
  "text=" + apoc.text.urlencode(a.text) + "&" +
  "lang=de&" +
  "pageRankSqThreshold=0.80&" +
  "applyPageRankSqThreshold=true&" +
  "nTopDfValuesToIgnore=200&" +
  "nWordsToIgnoreFromList=200&" +
  "minLinkFrequency=100&" +
  "maxMentionEntropy=10&" +
  "wikiDataClasses=false&" +
  "wikiDataClassIds=false&" +
  "userKey=" + $userKey as url
```

Once the Wikifier API request is created, the response of the request is fetched using the APOC method apoc.json.load. Additionally, the "UNWIND" clause is used again to convert the JSON data into annotation result objects.

```
CALL apoc.load.json(url) YIELD value
UNWIND value.annotations as annotation
```

In the next step, annotation results with a wikiDataItemId set to null are filtered out. For the remaining results, entity nodes are created. The wikiDataItemId is set as a unique identifier, with the title set as the title and the URL set as the URL of the node. By storing the Wikipedia URL and the wikiDataItemId as properties of the entity nodes, we can later easily

access additional information about the entities. The constraints also ensure that no nodes with the same wikiDataItemId are created multiple times.

```
WITH annotation, a
WHERE annotation.wikiDataItemId IS NOT NULL
MERGE (e:Entity{wikiDataItemId:annotation.wikiDataItemId})
ON CREATE SET e.title = annotation.title, e.url = annotation.url
```

Lastly, the "HAS_ENTITY" relationship is established between the article node and the newly created or already existing entity node, which was extracted from the text of the article.

```
MERGE (a)-[:HAS_ENTITY]->(e)
```

Classify entity nodes In this subsection, we will explore the process of classifying the recognized entities and organizing them into distinct nodes within our KG. We will discuss the methods and techniques employed for classification, as well as the steps involved in associating these classified entities with their respective nodes and relationships in our graph database. The class nodes can be utilized to gain deeper insights into the data within the graph and to enable more complex queries in the future.

In this step, we match all nodes in the graph with the label "Entity." The objective is to gather all entity nodes so that they can undergo additional processing.

```
MATCH (e:Entity)
```

At this stage, we take advantage of the fact that we have the wikiDataItemId as a property of the entity node. We construct a SPARQL query for the Wikidata API, designed to fetch the label (name) and class information for each entity node using its wikiDataItemId. Additionally, we apply a filter to obtain only German language results, ensuring that the labels and class information are consistent with the intended audience and scope of our KG.

```
WITH 'SELECT *
WHERE {
  ?item rdfs:label ?name .
  filter (?item = wd:' + e.wikiDataItemId + ')
  filter (lang(?name) = "de" ) .
  OPTIONAL {
    ?item wdt:P31 [rdfs:label ?class] .
    filter (lang(?class)="de")
  }
}' AS sparql, e
```

Here, we make a call to the Wikidata SPARQL API using the query constructed in the previous step. We leverage the APOC library function apoc.load.jsonParams to fetch the data from the API, which returns the response in JSON format. To process and extract the relevant results from the JSON response, we employ the UNWIND clause, allowing us to further analyze the information and integrate it into our KG effectively.

```
CALL apoc.load.jsonParams(
  "https://query.wikidata.org/sparql?query=" +
  apoc.text.urlencode(sparql),
  {Accept: "application/sparql-results+json"}, null)
YIELD value
UNWIND value['results']['bindings'] as row
```

At this point, we first verify if the API response row contains class information for the identified entities. If the class information is indeed available, we proceed to create a class node or match an existing one with the same name property as the class value retrieved from the API response. With the class node successfully created or matched, we then establish a relationship of type "INSTANCE_OF" between the entity node and its corresponding class node.

```
FOREACH(ignoreme IN CASE WHEN row['class'] IS NOT NULL THEN[1] ELSE [] END |
  MERGE(c: Class {name: row['class']['value']})
  MERGE(e) - [: INSTANCE_OF]->(c))
```

5.3.3. Example

In this subsection, we will explore an example of the KG, illustrating the structure and relationships between the nodes while also providing an overview of the graph's scale.

Figure 5.4 presents an example of a news KG constructed according to the previously described process. The graph is made up of an article node with the title "US Presidential Election", which is part of the news category "Politics". Further, the article has the tags "Elections" and "USA". NER and EL were used to extract the entities "Joe Biden," "Donald Trump," "Democratic Party," and "Republican Party" from the text of the article. The entity nodes were classified as a person and, as such, are "INSTANCE_OF" the class "Person". The two entity nodes "Democratic Party" and "Republican Party" have been classified as political parties and therefore are "INSTANCE_OF" of the class "Political Party".

After a specific example from the graph, some data about the general graph is presented below. The process of creating article, category, and tag nodes, as well as the creation of entity nodes and their classification was performed once per hour in the period from 15.02.2023 to 11.04.2023. The graph created through this process is used in the following.

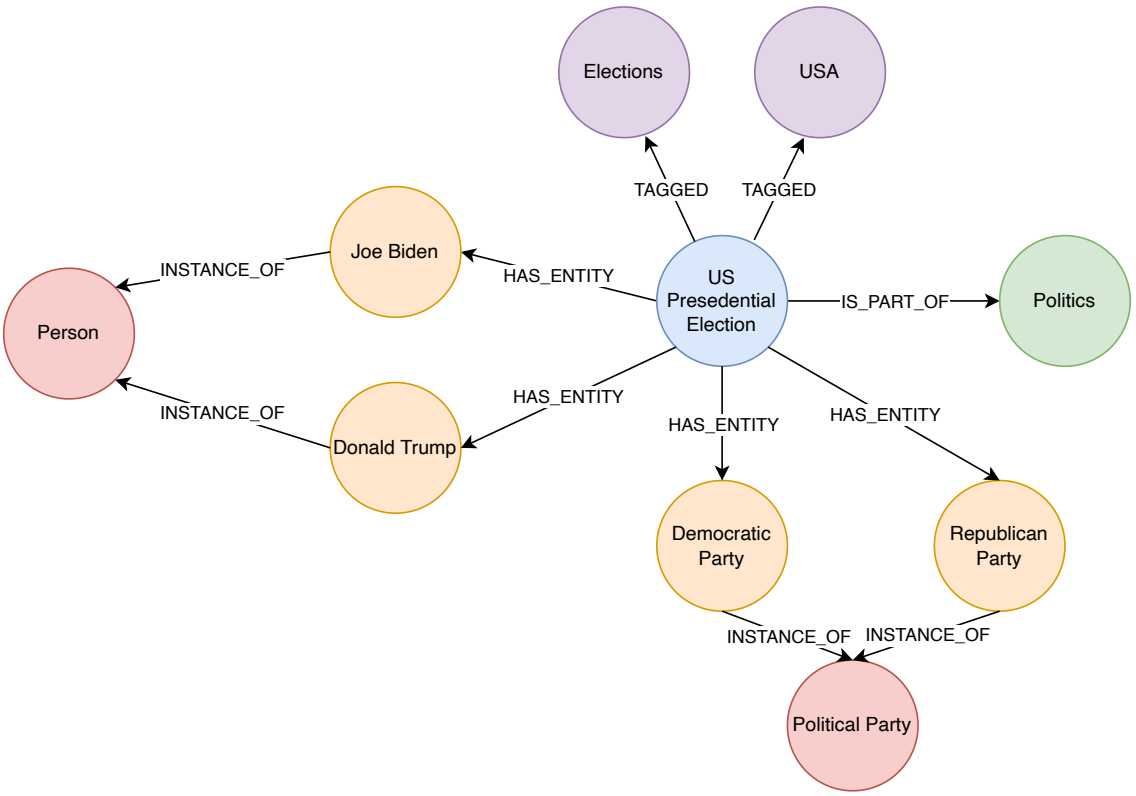


Figure 5.4.: News knowledge graph example

Table 5.3 depicts the composition of nodes of the exemplary German news KG, which consists of 8786 nodes. Of those roughly 24% of the graph, 2218 nodes in total, are article nodes. The article nodes belong to six categories, which make up only a fraction of the graph. Moreover, there are a total of 2524 tag nodes in the graph, accounting for 29% of the graph. Approximately one-third of the nodes in the graph are entity nodes as the graph is made up of 2524 entity nodes. Additionally, the graph contains 14% class nodes.

Table 5.3.: Nodes of exemplary German news knowledge graph

	Article	**Category**	**Tag**	**Entity**	**Class**	**Total**
Number	2118	6	2524	2874	1264	8786
%	24	0	29	33	14	100

Table 5.4 lists the relations of the exemplary German news KG. In total, the KG contains 25900 relationships. Of those relationships, eight percent are the "IS_PART_OF" relationship, which is between the article and category nodes. Based on that, it can be concluded that there are articles that belong to several categories. About a quarter (6775) of the relationships is the "TAGGED" relationship between article and tag nodes. This means that, on average, every article has roughly three tags. The "HAS_ENTITY" relationship is present the most often with

12567 relationships making up about half of the KG. It can be deduced that from each article about six entities can be extracted and also that each entity is extracted more than four times, on average. Lastly, the graph contains 4406 "INSTANCE_OF" relationships, which make up 17 % of the relationships. On average, each entity node is classified into 3.5 different classes.

Table 5.4.: Relationships of example German news knowledge graph

	IS_PART_OF	**TAGGED**	**HAS_ENTITY**	**INSTANCE_OF**	**Total**
Number	2152	6775	12567	4406	25900
%	8	26	49	17	100

5.4. Research question 4

In this section, we address Research Question 4: "How can we design and implement a voice-based German news search and exploration system using a knowledge graph?", by presenting the architecture and components of our conversational search agent, along with an overview of the voice interface and the connected webhook service that enables seamless voice-based interaction with the news KG.

5.4.1. System architecture

In this subsection, we will discuss the system architecture of our voice-based German news search and exploration system.

The system consists of a voice interface, a CA, a webhook service, and a KG. The search interface for the news KG is developed in the form of a voice-based CA. We used the NLU platform Google Dialogflow as described in chapter 2.1.3. The system architecture depicted in Figure 5.5 supports the dialogue turns in the following manner. A user initiates the conversation with the agent by clicking on a microphone button in der Web Application, whereupon the computer captures the user's utterance. Then, utilizing a speech-to-text model the input audio is transcribed into text and sent to the Dialogflow agent. Next, using an intent recognition model that was fine-tuned with sample utterances, the Dialogflow agent tries to predict the user's aim. At this point, the agent also performs entity recognition to extract entities contained in the user's query. Based on the identified intent and entities, the agent determines whether a standard answer, like a short salutation or a default answer, is sent back to the user or a more complex response is needed. In the latter case, a webhook request is sent to a Python-based webhook service, which extends the agent with a conversation fulfillment service. The webhook service forwards the request to the corresponding endpoint based on the request's intent and entity. Then, after having received the request, the webhook service processes it and matches it to a Cypher query template. Once the query template is completed with parameters from the extracted entities, it is executed to retrieve the information from the KG. Finally, for the response generation, the query result from the graph database serves as

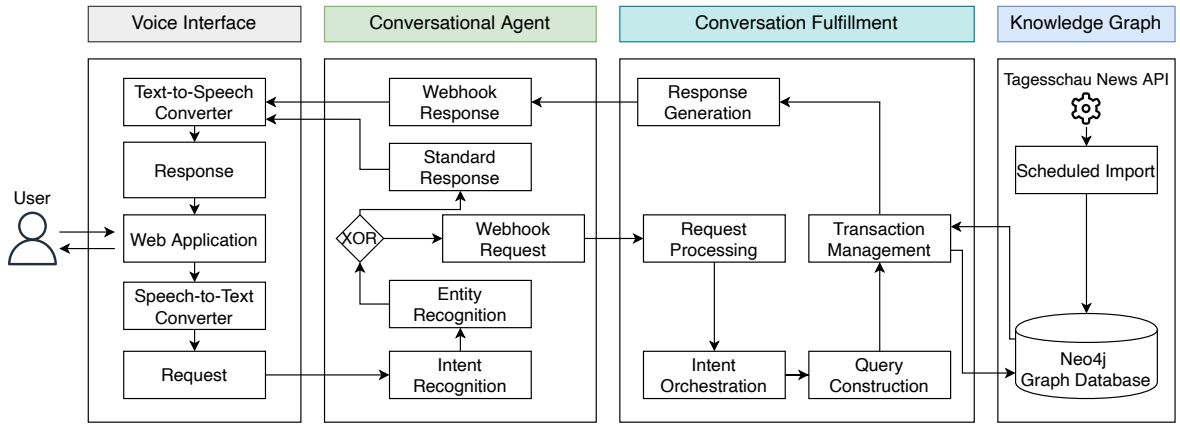


Figure 5.5.: Architecture of conversational search agent [91]

input. We construct the response by combining predefined textual elements and retrieved data items from the graph. The text response is augmented with speech Speech Synthesis Markup Language (SSML) to annotate paragraphs, and sentences, as well as to insert pauses and emphasis, thereby enhancing the comprehensibility. In the finishing step, to generate an audio file, the response is sent back to the agent which applies a text-to-speech model. This audio file is sent to the browser interface and is automatically played back to the user.

5.4.2. Voice interface

In this section, we focus on the voice interface, which serves as the main point of interaction for users and enables voice-based search queries and responses.

In Figure 5.6, a screenshot of the system's voice interface is displayed. We built a simple web application for the prototype of this thesis and to conduct a user study. Given the fact that the interaction with the system happens via speech in- and output, the web page shows only a button for recording audio and a short note for the user regarding the system's usage. When the system is waiting for input from the user, the message "Press to speak" is displayed as a notice. After the user presses the button and the system is ready for voice recording, the message changes to "Please speak". Thereafter, when the user has finished speaking and the request is processed by the CA and the webhook service, "Please wait" is visible. As soon as the answer is sent back to the frontend, "Press to speak" is shown again, since the user can interrupt the voice output and enter a new voice command at any time, even while the answer of the previous question is still being played back, by pressing the button again. As a visual aid, the button pulses and casts a colored shadow, during voice recording.

The web application was developed using Python Flask [1], a lightweight web framework specifically designed for rapid web application development and APIs. It is based on the

[1]Python Flask: https://flask.palletsprojects.com/

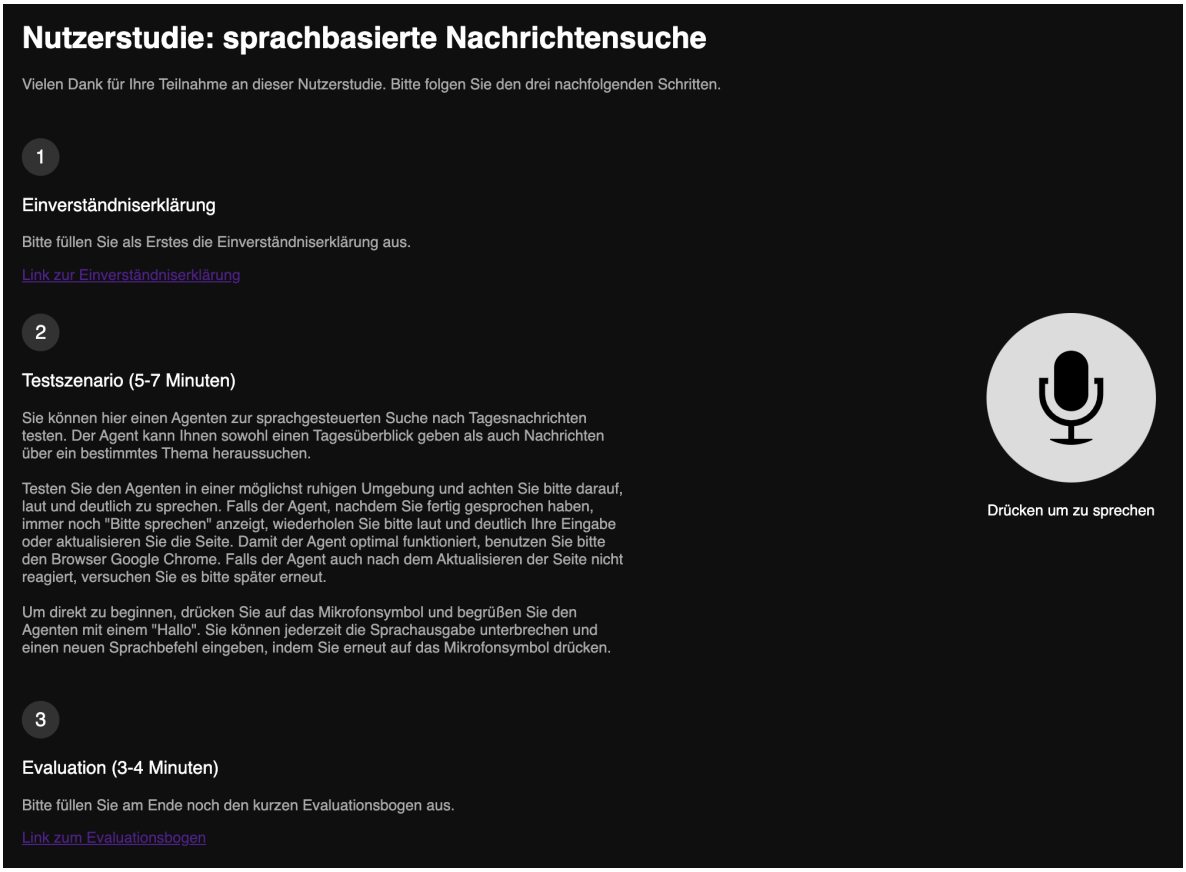


Figure 5.6.: Screenshot of voice interface

WSGI toolkit and Jinja2 template engine and provides a flexible structure to connect the frontend with the required backend services.

To enable speech input functionality within the web application, we used the Web Speech API [2]. It is a JavaScript-based interface that provides web applications with the ability to use speech recognition and speech synthesis. The API consists of two main components: speech recognition and speech synthesis. The first component allows the application to convert spoken language into text by accessing the speech recognition services of the browser or operating system. This enables accurate transcription of user input in real-time. The second component, speech synthesis, converts text to spoken language by accessing the speech synthesis services of the browser or operating system. This allows text-based responses from the system to be played back as audio output that is easy for the user to understand. This thesis prototype utilizes the Web Speech API only for speech recognition in order to recognize the speech of the user and to recognize when the user finishes his speech input.

[2]Web Speech API: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/

The user must first press the microphone button in order to start the interaction and to speak his voice input. The Web Speech API automatically detects when the user stops speaking, and then stops recording. In the case that the user stops speaking but, for example, due to poor internet connection or background noise, the Web Speech API does not detect that the user has stopped speaking, the recording will also stop after 1.5 seconds without new voice input. As soon as the recording is finished, the text recognized by the Web Speech API is sent as a request to Dialogflow, which responds with an audio output. This is converted to an audio file in the frontend and played automatically for the user without using the Web Speech API.

5.4.3. Dialogflow agent

In the following, we will examine the core component of our conversational search agent namely the Dialogflow agent.

As described in the architecture overview 5.4.1, the agent receives the user's request, which is speech input, converted to text by the frontend. To determine the user's intent, the agent then uses an intent recognition model fine-tuned with sample utterances. In addition, entity recognition is performed to extract the entities from the user's query.

In Dialogflow, "intents" and "entities" are vital for understanding and processing user requests. Firstly, an "intent", such as "news.search" or "help" represents the user's goal or intention behind a particular speech input and varies depending on what action the user wants to perform with his request. By using intents, user requests can be categorized and appropriate actions can be triggered. Each intent is trained with specific training utterances to improve recognition accuracy and ensure that the agent correctly interprets user queries. Secondly, a Dialogflow "entity" is a crucial piece of information that is identified and extracted within a user request. In short, Dialogflow "entities" allow the agent to understand specific information from a user request and use that information for further processing or to perform actions. Given the exemplary user query "show me yesterday's news from Berlin", examples of entities extracted from it can be "date" and "location", respectively. Dialogflow "entities" can also be categories like "economics".

The CA was trained to recognize multiple intents encapsulating various user requests. In Table 5.5 all intents with exemplary user utterances for which the agent matches the intent and the agent's answers are displayed. They serve as training sentences for this thesis' prototype. The intents are based on the scheme of the interaction pattern elaborated in section 5.2. The chapter also describes the heuristics used to select the articles or entities for the intents. The italicized words in the training sentences represent the entities that are extracted from the utterances.

The first intent in Table 5.5 is the "default.welcome" intent, which is detected when the user greets the agent. In response, the user is greeted back and the conversation is initiated.

When the user asks for help, the user gets assistance and example instructions for dealing with the system. Since the user is asked at the end of the response if he needs any further help, there are the "help.yes" and "help.no" intents. The "news.search" intent is matched when the user's query is about news in general. The user is then asked whether he prefers a daily overview or news about a specific topic. We divided the news search intents into three levels of abstraction, according to the question catalog in Table 5.1. Firstly, if users request a daily overview, the agent selects the "news.overview.search" intent, providing them with three articles from the domestic, foreign, and economic categories. Secondly, users can search for news within a specific category, triggering the "news.category.search" intent. The category is extracted from the user utterance. The user's statement is assigned to the "news.search.entity" intent when he directly queries news about a specific entity. Again, the entity of interest is extracted from the user's statement. In both cases, following the heuristic presented in section 5.2, three articles related to the extracted Dialogflow entity are returned. The "news.category.list" intent is triggered when users desire news on a particular topic without mentioning it, prompting the heuristic from section 5.2 to suggest three example categories and three example entities. If the user requests additional articles, their statement is assigned to the "news.suggest.search" intent, providing them with three more articles about the previously selected overview, category, or entity, respectively.

To enable users to select an article from the available options, the "news.select.article.by.number" and "news.select.article.by.keyword" intents are provided. In the former, users select the article based on its position in the sequence, with the "sys.number" system entity employed to extract the number from the statement. In the latter intent, users mention a portion of the article's content, and the system extracts that part using the "sys.any" system entity. In both cases, the agent forwards the selected article to the frontend, including the previously explained related entities of the article. The control intents allow users to manage the news articles in the selection. If users want to hear the next article, the "control.next.article" intent is triggered; for the previous article, the "control.previous.article" intent is activated, and the "control.repeat.article" intent is used to repeat the article. The control.stop intent is triggered if users wish to interrupt the conversation. In the event of unexpected inputs that the agent cannot comprehend, a fallback intent is activated. Although the agent possesses a finite set of intents, the news content within the graph, which changes daily, leads to a virtually infinite number of potential conversation paths.

Contexts are another key element in Dialogflow, serving to manage conversations and store information during user interactions. They enable the agent to maintain the state of a conversation across multiple dialogue turns and access relevant information gathered in previous dialogue rounds. As a result, Dialogflow can deliver coherent and cohesive responses based on prior information. These concepts assist Dialogflow in better understanding user needs and providing appropriate responses or actions.

Table 5.5.: Dialogflow agent intents

Intent name	**User utterance**	**Agent response**
default.welcome	Hi Good Morning	Greeting and a request example of how to ask for news
help	I need help. What can I ask you?	Responds with some example requests and some context about the agent
help.yes	Yes I want more help. Give me more help.	Presents the user more example requests
help.no	I do not need more help. No, thank you!	Reminds the user how to ask for help again
news.search	What is the news? Tell me the news	Counter-question, whether the user rather wants a daily news overview or news about a specific topic
news.overview.search	I want a short Overview Give me a brief summary	The latest articles from domestic, foreign and economy news
news.entity.search	News about *Donald Trump*. Do you have news about inflation?	Returns the three most recent articles about the requested entity
news.category.search	Give me news about *politics*. I want science news.	Returns the three most recent items from the requested category
news.category.list	What are the categories? What news do you have?	Lists categories and topics about which the agent has news to report
news.suggest.search	More headline suggestions Can you show me more articles?	Three other recent articles related to the previous topic
news.select.article .by.number	Read me the *second* article. I want article number *3*.	Reads out the requested article and suggests articles to similar topics at the end
news.select.article .by.keyword	Read the article on *climate change*. The article about *Nord Stream*.	Reads out the requested article and suggests articles to similar topics at the end
control.next.article	Next Skip article	Next article from selection
control.previous.article	Back Previous article	Previous article from selection
control.repeat.article	Repeat Read article again	Repeats current article
control.stop	Stop	Stops the conversation

In our prototype, we also employ contexts to facilitate Dialogflow's matching of the correct intent based on both the user statement and the dialogue context. Firstly, we utilize the "help-followup" context after triggering the help intent to correctly match the "help.yes" and "help.no" intents. The context is set with a lifespan of one, ensuring its validity only for the next voice input. This prevents the utterances "Yes" and "No" from automatically triggering the "help.yes" or "help.no" intent during the dialogue, allowing them to be matched exclusively following the help intent.

Another context is the "select" context, which is set after presenting three articles for the user to select from. This serves, for example, to avoid confusion between "Read me articles about Donald Trump" when the user wants articles about Donald Trump, and "Read me the article about Donald Trump" when the user wants to select an article from the presented choices with Donald Trump in the headline. When the "select" context is set, the intent "news.select.article.by.keyword" is matched. Otherwise, the intent "news.entity.search" is matched. The "select" context is also set with a lifespan of 1.

As the agent is designed for exploring German news, the language of the Dialogflow agent is set to German. To make the agent's voice as pleasant as possible, we employ the "de-DE-Wavenet-B" voice with a speaking rate of 1.05.

Fulfillment in Dialogflow allows for the creation of complex and dynamic responses to user inquiries based on recognized intents and extracted entities. The agent uses a webhook to access external resources such as our KG and provide custom responses. This enables the agent to deliver precise and context-specific information from the KG to users. The functionality and implementation of the webhook service are elaborated in the following subsection.

5.4.4. Webhook service

The webhook service is a central component of the system, serving as an interface between the Dialogflow agent and the KG. In this section, we explain the key aspects and functionalities of the webhook service, including special features for processing user queries, logging interactions in MongoDB, and providing an overview of the associated code repository structure.

For the implementation of the webhook service, we again relied on a Python Flask application. We used the "Flask Dialogflow" package [3] to realize the webhook service within our Flask application. By employing "Flask Dialogflow," we benefit from a familiar Flask extension structure, enabling us to implement the webhook service efficiently while integrating the Dialogflow agent into the Flask application. The package ensures requests are automatically forwarded to the appropriate webhook for the defined intent and facilitates sending the response back to Dialogflow.

[3] Flask Dialogflow: https://github.com/ONSEIGmbH/flask-dialogflow

The webhook service receives requests from the Dialogflow agent and generates responses based on the entities and intents contained in the requests by accessing the KG and inserting the retrieved data into predefined templates. Upon receiving a request, the service first constructs the appropriate query to obtain the required information from the graph. One example is the following query, which returns the titles of articles containing a specific entity, sorted by descending publication date. In this case, the variable "$entity" is undetermined and replaced by the user-requested entity, which could be Donald Trump, for example. The assembled query is then sent to the Neo4j KG.

```
MATCH (a:Article)-[:HAS_ENTITY]->(e:Entity {title: $entity})
RETURN a.title AS title
ORDER BY a.date DESC
```

After querying the graph, the service generates the response by inserting the retrieved data into a response template. In the above example, if no articles about Donald Trump are found, the template "Unfortunately, no news about "$entity" could be found." would be filled with the entity Donald Trump, resulting in the response "Unfortunately, no news about Donald Trump could be found."

To give the agent's responses a more natural sound, SSML elements are integrated into the templates. SSML is an XML-based markup language designed to enhance the quality of speech synthesis. With SSML pronunciation, pitch, speech rate, and volume of the synthesized voice can be adjusted precisely. An example of this is the following sentence:

```
<speak> <s> For a daily overview, say "overview" or say <break time="400ms"/> news on a
↪  specific topic. <s> </speak>
```

In this example, the <speak> tag indicates the use of SSML. The <s> tag signifies a sentence, while the <break time="400ms"/> tag inserts a brief pause to clarify better the various options within the enumeration for the listener.

A distinctive feature of the response templates is the integration of exemplary requests at the end of each response. This makes it easier for the user to choose the next voice command in a way that the system can process and presents possible options for the next query. For instance, if three articles about Donald Trump are presented, the following is added after the article titles: "For example, to select one of the articles, say: 'Read me the second article', or for more suggestions, say: 'More suggestions'." This clarifies for the user how to select articles or request additional suggestions.

A specialized process in the webhook service involves determining the correct entity in the graph when a user asks for news about an entity. Since the user may employ various

spellings that do not necessarily match those in the graph, so-called fuzzy matching is used to identify the entity with the highest similarity score. This is also applied when the user selects an article not by number but by keywords or content. In this case, the user's statement is compared only to the suggested article titles to select the best-matching article. In both instances, the threshold for the minimum similarity rating is set at only 0.2 to minimize returning error messages to the user. However, this low may lead to a high rate of mismatch. If no entity is recognized, the user is simply informed that no news regarding the topic of interest can be found. If no article can be confidently selected, the user is asked to clarify their choice more precisely.

To store additional information related to the Dialogflow context and log user statements along with further information, we employ a MongoDB[4]. MongoDB is a document-oriented NoSQL database designed for high scalability, flexibility, and performance. Unlike relational databases, MongoDB is based on a hierarchical structure of documents and collections for data storage. Each document is stored in Binary JSON (BSON) format, facilitating the storage and querying of data in JSON format. MongoDB is ideal for applications with variable data structures or large data volumes.

In MongoDB, we store the last requested category, the last requested entity, the order of the last selected article, and the three most recently presented articles. This enables navigation commands and the ability to request additional articles related to the previously mentioned entity or category. MongoDB also serves as a logging tool. For each user statement, a timestamp, the user's utterance, and the recognized intent are stored. In a similar manner, the agent's response, a timestamp, and the associated intent are logged. These logs enable better analysis of the system's behavior in user studies and the identification of opportunities for improvement.

The structure of the code repository is explained below to provide a better understanding of the implementation. The code repository includes:

- Main application file: This file contains the Flask application, the integration of the Dialogflow agent, and the connections to the KG and the logging database.
- Webhooks: The "webhooks.py" file defines all endpoints for individual intents and forwards the request to the appropriate handler function.
- Intent Handlers: The "handlers.py" file contains all specific handler functions for the various intents, where most of the logic lies.
- Helper Functions: The "controllers.py" file includes reusable functions used in the webhook service, such as functions for retrieving data from the KG, processing user requests, or generating responses.

[4]MongoDB: https://www.mongodb.com/

- Templates: "queries.py" contains all Cypher queries that can be adjusted according to requirements. "responses.yaml" stores all response templates into which data from the KG is inserted.
- Graph Population: The script in "fetch.py" is executed hourly to fill the graph with data.
- Configuration Files: These files contain settings and configurations for the application, such as the "requirements.txt" file that lists all packages used for the webhook service.

In view of the user study discussed in the next section, the webhook service is a central component of the system. It enables the understanding and analysis of interactions between users and the Dialogflow agent. Insights from the user study contribute to optimizing the implementation of the webhook service and thus improve the overall quality of the system.

5.5. Research question 5

In this section, we address the final research question, Research Question 5: "*Which insights can be gained from user tests for improving the conversational agent?*" To answer this question, we conduct practical tests on the developed system, subsequently analyzing the insights to enhance the performance of the CA.

5.5.1. Initial user test

Before the final comprehensive evaluation study, we conducted an initial user test with a sample of eight participants. The aim of this user study was not only to test the basic functionality but also to identify technical and non-technical issues. The age range of the participants was between 21 and 54 years, with an average age of 33.6. The gender distribution was 37.5% female and 62.5% male. No specific instructions were given to the users, as we wanted to investigate how intuitively understandable the agent is.

After the participants had tried out the system, they answered some questions about their experiences. There were both open-ended questions and those with a Likert scale from 1 to 5, where 1 was the most negative and 5 was the most positive. Overall, participants were not averse to the system, with an average score of 3.25. They were even more satisfied with the informational content of the responses, which averaged 4.125. They were also satisfied with the system's usability, which also scored an average of 4.125. Users were satisfied with the speed of speech output and the scope of responses. In general, users appreciated the innovative concept of voice-based exploratory news search and praised its user-friendliness. No one had difficulty in understanding the core functions of the system. Participants could obtain both a brief general overview and targeted information on specific topics that had piqued their interest. It became evident that users quickly learned to use the entity-based search. They successfully searched for news about numerous entities, such as the bank Credit Suisse, the person Mark Zuckerberg, and the country Ukraine.

Although starting the news-seeking dialogue was not problematic for users, some suggestions for improvement were mentioned. Some participants experienced technical issues with certain browsers, which affected microphone usage and system stability. These and other problems led to the agent not understanding all user utterances. Regarding the suggested article headlines, test users preferred a numerical enumeration over a simple list. This way, they could select an article by statements like "first" or "second article." However, other users wanted the option to choose articles by mentioning only a keyword from the article's headline. Furthermore, some participants expressed a desire to customize the speech output. They wanted to adapt the voice style or reading speed to their needs, for example. Some users also found the agent's voice to be annoying or disruptive.

The results of this initial user test have been incorporated into the agent's further development, helping to address both technical difficulties and user preferences. The insights gained from this test enabled targeted optimization and improvement of the system to achieve higher user satisfaction and more effective information retrieval through the voice-based news exploration system.

5.5.2. System evaluation results

Following our initial evaluation and the improvement of the system we conducted a comprehensive, extensive user study to further assess the effectiveness of the developed system. The primary objectives of this final evaluation were to uncover any technical limitations, understand the practical usage of the conversational search system for news exploration, and identify areas for improvement.

Study sample In this study, we obtained a diverse sample of participants through university courses, friends, and social media. The participants were not given any instructions on how to use the agent or what news was available, as we wanted to test the self-explanatory capabilities of the agent. The only instruction given was to greet the agent and be aware that the agent serves for voice-based daily news search. After testing the agent, users were asked to report their experiences in an online questionnaire.

Table 5.6 provides a breakdown of the user study participants. A total number of 54 participants took part in the study. The gender distribution was almost balanced, with 42.5% female and 57.5% male. Participants' ages ranged from a minimum of 14 to a maximum of 86 years, with an average age of 36 years. The largest group of participants (61%) was between 14 and 30 years old, with most being young adults between 20 and 30 years of age. Only five participants lived outside of Germany. Most participants (19) lived in Bavaria, 12 in North Rhine-Westphalia, with the remainder distributed evenly across Germany. Additionally, 65% of the participants were single. The household situation was balanced, with most participants living with their partners, followed by those living alone and those living with roommates or their parents. The sample demonstrates a high level of education, with 35 of the 54

participants having at least a university degree (Bachelor, Master, or Ph.D.).

A clear trend emerged in news consumption per week: 55.5% of participants consumed news daily, with a further 28% consuming news 4-6 times per week. Almost all participants (50) consumed their news digitally, 29 still watched the news on television, and only 10 read a physical newspaper. Although 76% of participants had prior experience with voice assistants, only 2 of the 54 participants used them for news consumption. This indicates that the study was conducted with a diverse sample across different age groups and genders, with no one having used CAs for news search before. Therefore, the sample is ideally suited for evaluating the CA.

Conversation log analysis The 54 users conducted a total of 66 dialogues with the system, as some of the users engaged in multiple dialogues with the system. There were a total of 2174 conversation turns, with a turn being a statement from either the user or the agent. On average, a session consisted of 32.9 turns. A dialogue session lasted an average of 9 minutes. The longest conversation with the agent lasted a full 41 minutes, and the shortest was just one second. Due to some very extensive and very short interactions with the agent, there is a significant standard deviation of 7.3 minutes. In the following, we analyze the dialogues from the conversation logs.

In Figure 5.7, the most important matched intents of each user utterance are depicted. There were 192 matches with "default.fallback", thus 82% of utterances could be matched with one of the intents. The "default.welcome" intent was matched 77 times. This is more than the 66 conversations, as some participants greeted the agent several times. Besides the 77 greetings, the agent was only asked for help 40 times. With 222 entity-based news searches, 48 overview searches, and 18 category searches, the agent was asked for news a total of 288 times, with the entity-based search clearly standing out at 77%. Across all three types of search, additional article suggestions have been requested a total of 72 times. Using a keyword or sequence number, a total of 152 articles were selected. In the article selection, there was navigation a total of 82 times, with 58 times either the next article being selected or the current one being skipped. The current article was repeated only 12 times and the previous article was selected only 12 times. This means that a total of 216 articles were consumed. Participants had categories and entities suggested 31 times.

Given the prominence of the entity-based search intent compared to other intents, and its central role in this study, it is subject to a more detailed analysis. Figure 5.8 illustrates the most frequently searched entities and their corresponding classes. From the left side of figure 5.8a, it is apparent that users most frequently queried the entities Ukraine, China, Olaf Scholz, Football, and Taiwan. On the right side of figure 5.8b, the classes of the searched entities are displayed. Countries, with 63 searches, represent the most frequently occurring category. This is followed by 27 searches for persons and 23 searches for various other miscellaneous classes. Additionally, news about cities and sports were also frequently requested.

Table 5.6.: Characterization of participants

Category	**Options**	**Population (n = 54)**	**Percentage (%)**
Gender	Male	31	57.5
	Female	23	42.5
Age group	14 - 30	33	61
	30 - 60	17	31.5
	60 and above	4	7.5
State of residence	Baden-Württemberg	3	5.5
	Bavaria	19	35.5
	Berlin	5	9
	Hamburg	7	13
	Hesse	2	3.5
	Lower Saxony	1	2
	North Rhine-Westphalia	12	22.5
	Not in Germany	5	9
Marital status	married	16	29.5
	single	35	65
	divorced	3	5.5
Household situation	Living alone	16	29.5
	Living with partner	25	46
	Living with parents	3	5.5
	Living with roommates	10	18.5
Educational status	No degree	2	3.5
	Abitur	14	26
	Apprenticeship	3	6
	Bachelor's Degree	11	20
	Master's Degree	17	31.5
	PhD	7	13
News consumption per week	0	4	7.5
	1 - 3	5	9
	4 - 6	15	28
	7	30	55.5
News consumption	Newspaper (physical)	10	18.5
	Online (digital)	50	92.5
	Radio	19	35
	Television	29	53.5
	Voice assistants	2	3.5
Experience with voice assistants	Yes	41	76
	No	13	24

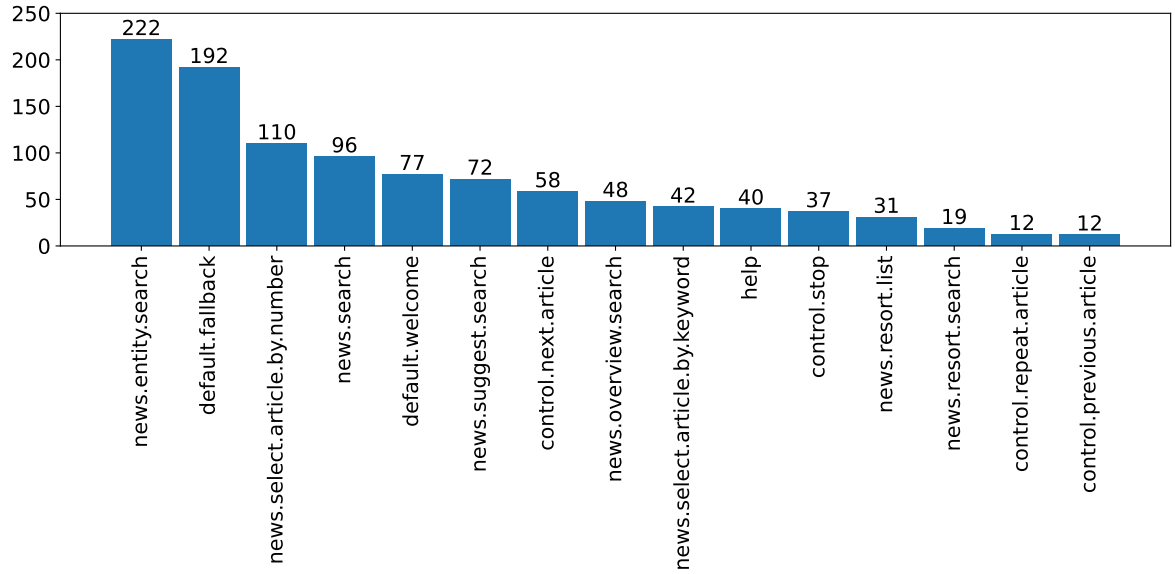
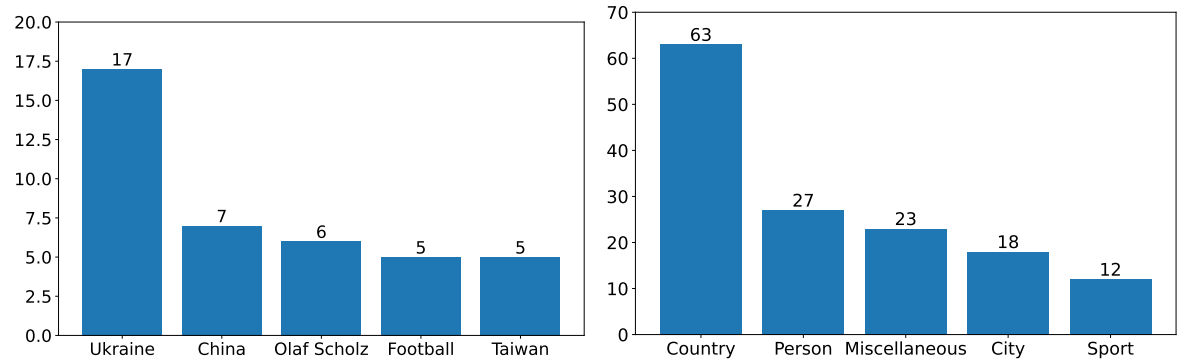


Figure 5.7.: Distribution of matched intents



(a) Top searched entities

(b) Top classes of searched entities

Figure 5.8.: Entity-based news search entities

However, not all entities requested by the users were correctly recognized by the system. Of a total of 222 entities requested, 58.5%, equivalent to 130 entities, were recognized correctly. Table 5.7 provides an overview of which classes were recognized most frequently and which were not recognized often. Among the recognized entities, 40.8% or 53 entities belonged to the class "Country", which had a matching rate of 93.5%. The second most common class was "Person", which accounted for 14.6% of the recognized entities and was identified correctly 19 times, with a matching rate of 70%. Entities from the "City" class were also recognized frequently, with a matching rate of 78%. In total, an entity from this class was recognized correctly 14 times. About a third or 44 of the recognized entities, belonged to various smaller classes. On the other hand, 92 of the 222 queried entities, which corresponds to 41.4% of the

entities, were not recognized correctly. Entities from the "Sport" class were most often not recognized correctly, which had a low matching rate of 17.4%. Eight entities from the "Person" class were also not recognized correctly, however, these were relatively reliably recognized with a matching rate of 70%. About 7.6% of the not recognized entities belonged to the "Nutrition" class, which also had a low matching rate of 28.6%. Of the 92 entities not correctly recognized, 67 belonged to many different classes.

Table 5.7.: Accuracy of linking recognized entities to the knowledge graph

Entity classes	**Count**	**%**	**Matching rate**
Recognized entities	130	58.6%	
Country	53	40.8%	93.5%
Person	19	14.6%	70.0%
City	14	10.8%	78.0%
Rest	44	33.8%	
Unrecognized entities	92	41.4%	
Sport	10	10.9%	17.4%
Person	8	8.7%	70.0%
Nutrition	7	7.6%	28.6%
Rest	67	72.8%	

The recommendation of entities related to the article also proved effective. After being presented with an article, users decided to request news about an entity in 38 cases. In 17 of these cases, one of the suggested entities was chosen, representing a ratio of 45%. To illustrate entity-based news search using the recommendations, we have depicted a representative dialogue from the user study logs in Figure 5.9. On the left side, the matched intent of the user utterance is visible. The user's statement in green in the middle column is paired with the system's response in gray. The right side visualizes the nodes of the KG, with the entity nodes represented in yellow and the article nodes in blue. The arrows between the individual nodes symbolize the user's path through the KG. To simplify visualization, not the entire KG is depicted and the nodes are not arranged in a complete graph structure, but rather so that it is clear which nodes the system uses for the response in each step.

In the beginning, the user asks the system for news about France, which matches the intent "news.entity.search". In response, the system provides three articles about France. In the next step, the user selects the third article titled "Building collapsed in Marseille". After the system has read out the selected article, the user desires more articles. The system then reads out three additional articles that are related to the entity France, in accordance with the intent "news.suggestion.search". The user opts for the second article. After the system has read out this article, it suggests three related entities: Ukraine, Russia, and European Union. The user selects Russia and from the articles suggested by the system, he decides on an article about EU sanctions for the Wagner Group. Following the reading of the article, the relevant entities Germany, Russia, and Ukraine are proposed. The user chooses Ukraine and from the

three suggested articles, he selects the third. At the end of the article, he navigates to the next article from the selection by saying "Next article". This command corresponds to the intent "control.next.article", and he repeats this step once more. Upon reaching the final article from the selection, he is again suggested three related entities at the end of the article's reading: Putin, Russia, and Ukraine. This time, he decides on Putin. The three suggested articles about Putin are no longer depicted and the conversation continues, not fully represented until the end.

Evaluation form For the evaluation, users were given a questionnaire with various questions, after they tested the system. In addition to several free-text form fields where they could describe their experiences, Likert-scale questions were also asked, using the same scale as in the initial user test. The free-text questions covered users' experiences, which news items were not available in the KG, their suggestions for improvement, and their desired features for the system. The Likert-scale questions focused on general satisfaction, the informational content of the answers, control over the news consumption, the suggestions provided by the voice assistant, and the understanding of the voice assistant. These questions included how easy it was to get the voice assistant to do what the user wanted, how understandable the answers were, and how comprehensible the behavior of the voice assistant was. Furthermore, users were asked about how human-like their interaction with the system was perceived. The scope of the answer and the speed of the speech output were also inquired again. At the end of the questionnaire, the standardized questions for the System Usability Scale (SUS) were asked. The full questionnaire is provided in the appendix A.1.

Compared to the overall satisfaction in the initial user study with a value of 3.25, the overall satisfaction in this study increased to an average of 3.7. Although this is not directly comparable, as there were intentionally no participants who took part in both studies. Users rated the informativeness of the answers as above average, with a mean score of 3.8. The ease of controlling news consumption was rated slightly lower, with a mean score of 3.65. The suggestions made by the voice assistant, in turn, received an average score of 3.8. It is worth noting that the comprehensibility of the system's answers was particularly well-received, with a mean score of 4.4. Users rated how easy it was to get the voice assistant to do what they wanted and the general behavior of the agent with average scores of 3.5 and 3.6, respectively. The worst score was for the question regarding the human-like behavior of the agent, with an average of only 2.8. However, users were satisfied with the speed of the speech output and the length of the responses.

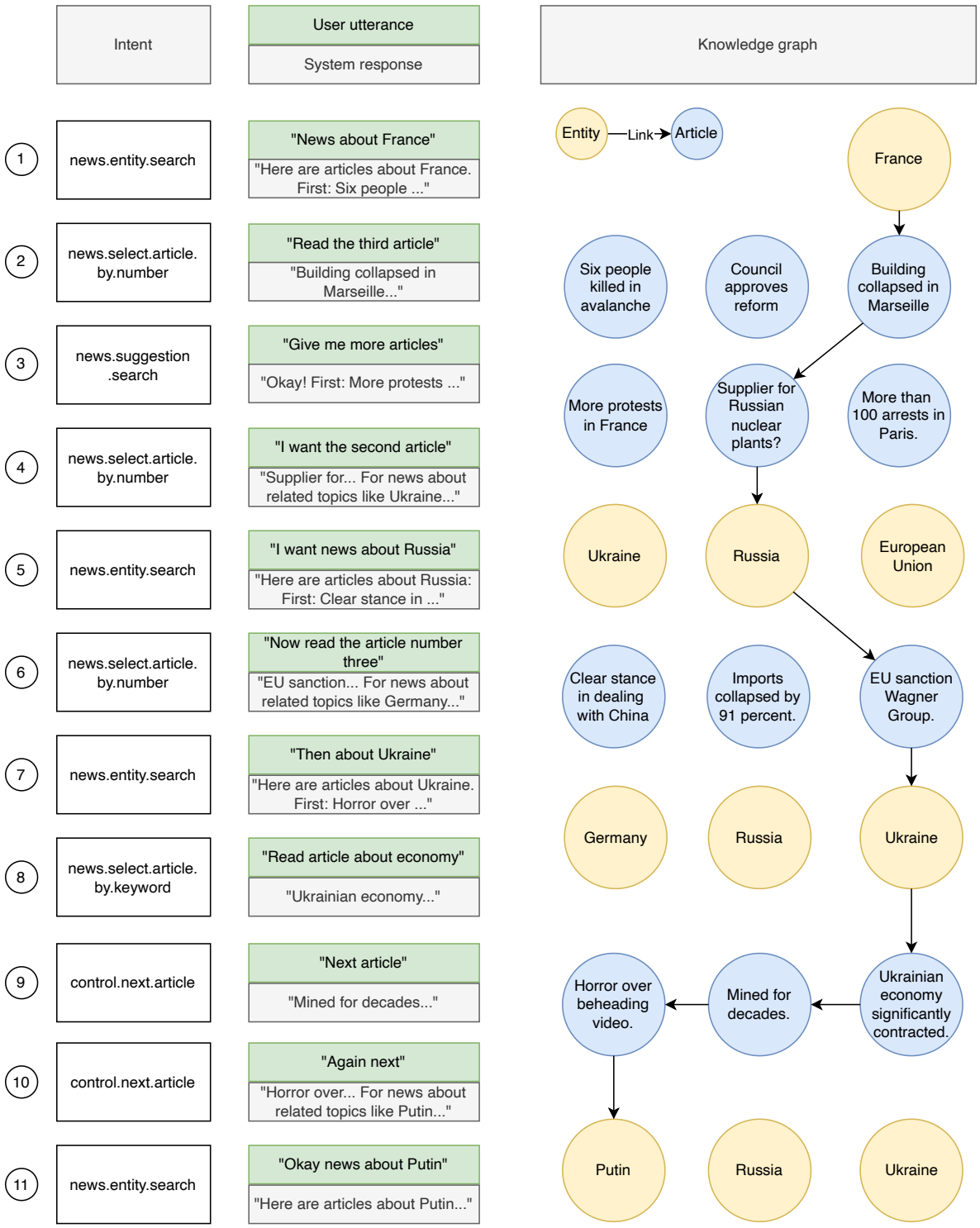


Figure 5.9.: Entity-based news search conversation example

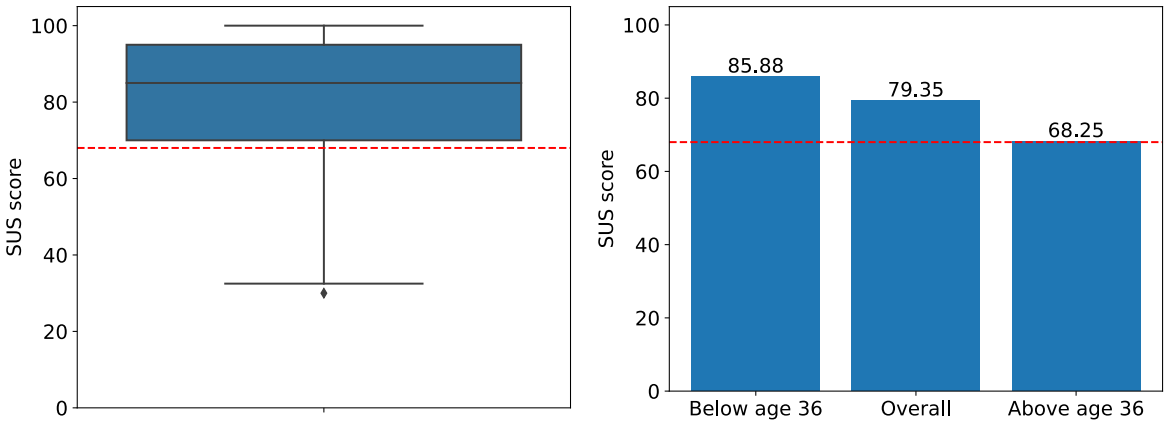
Correlational analysis During the analysis, several correlations were identified. For instance, a correlation between the relevance of the information perceived by the users and their overall satisfaction was observed. Furthermore, overall satisfaction correlated with control over the news flow and the suggestions provided by the voice assistant. There is also a strong correlation between the system's usability and overall satisfaction. Users carefully read the questions and had no problems understanding them, as no unexpected correlations or outliers were detected. It is also worth noting that overall satisfaction decreases with age, as older individuals had more difficulties using the system than younger people, although older people consume more news than younger individuals.

System Usability Scale The SUS is a well-established and reliable tool for assessing the usability of products, systems, or applications [95]. Developed by John Brooke [95], this 10-item questionnaire offers a quick and effective method to gauge subjective user satisfaction. Each question provides five response options, ranging from "Strongly disagree" to "Strongly agree," and the resulting SUS score is calculated using a specific formula, yielding a score between 0 and 100 [95]. Higher scores signify better usability, with a score of 68 typically considered as average usability. It is crucial to interpret the SUS score within the context of the specific application and user population. Comparing SUS scores between different systems or iterations can reveal valuable insights into their relative usability [95]. Since our user study was conducted in German, we used the translated questions from Rummel [96].

In Figure 5.10a, a box plot is displayed, which shows the SUS of our system based on the participants' responses. The overall score was 79.35 with a standard deviation of 18.47. This value is significantly above the average of 68. It is apparent that users generally found the system easy to use, as more than 75% of users reported an above-average SUS score. The correlation analysis has already revealed that older people had difficulties using the system. This behavior is also reflected in the SUS scores. As seen in Figure 5.10b Individuals below the average age of 36 had a SUS score of almost 86, which is significantly higher than those above the average, who had a score of only 68.25. Nonetheless, this score is still slightly above the average. The red dashed line in both figures indicates the general average SUS value.

In addition to the Likert scale questions and the SUS, we evaluated the free-text questions concerning user experience, suggested improvements, and desired features. 13 participants indicated that they found the agent's article and topic suggestions interesting. 14 participants expressed positive sentiments about the broad range of news available on various topics. Control over news selection and overall system control was particularly emphasized by 12 participants. Four individuals highlighted the convenience of not having to type, but being able to simply speak. Five participants appreciated the feature of having news read out to them instead of having to read it themselves, emphasizing that this was particularly beneficial for older individuals.

However, problems were also reported. For instance, 19 participants complained about



(a) SUS overview

(b) SUS comparison by age group

Figure 5.10.: System Usability Scale (SUS)

not receiving the correct news in response to their requests. Eleven individuals mentioned that the agent did not understand them when they used varied expressions; free speaking, in particular, did not function optimally for them. Twelve participants criticized the agent's pronunciation, which was at times very mechanical and especially mispronounced anglicisms and abbreviations. Four individuals found it disturbing that repetitive sentences often appeared in the responses during prolonged system usage. Three participants mentioned that usage was only possible in a quiet environment. Additionally, seven participants reported technical issues, such as a malfunctioning voice recording.

Regarding desired features, six users expressed a desire to adjust the speed of the voice output during usage. Six users would like to read subtitles while the text is being spoken. Three users would appreciate it if the agent, when not understanding them, would provide suggestions about news or voice commands as a response. Seven users desired a brief daily summary containing many articles in quick succession and covering all news categories. Eight users wanted news from additional categories, such as regional news, fashion, football, or music, not just the classic headlines. Two users expressed a wish for content other than news, such as weather or general information, to be included. Five users expressed a desire to query more information about the news sources and to be able to choose these themselves. Lastly, six users expressed a desire for the system to be integrated into existing systems, such as Alexa, their car, or a proprietary app.

6. Discussion

6.1. Key findings

The central goal of this study, "Can we use knowledge graphs to develop a conversational agent for German news search and exploration?" was addressed by implementing a German news knowledge graph and a conversational agent, and validating them in a user study. The user study, conducted with a balanced and diverse sample of 54 participants, generally showed that this question can be answered affirmatively. The log analysis revealed that a total of 66 dialogues with a total duration of 10 hours provided a sufficient amount of analysis material. The following discussion will elaborate on the results presented in Chapter 5 and draw conclusions on the approach and implementation, to determine their validity and potential for improvement.

The implementation was adequately functional, allowing users to utilize the system. This is supported by the high intent matching rate of 82%, demonstrating that the system reliably responded to users. Therefore, research question 4 concerning the fundamental implementation is answered. The majority of users found the core function of voice navigation for news useful and would like to use the system more frequently. Users also appreciated the idea of voice-based news search, as it obviated the need to read or type. Furthermore, the agent's responses were rated as understandable, and its behavior was deemed comprehensible. Users not only rated the agent's responses as understandable but also informative. The System Usability Scale (SUS) confirmed the general usability of the system through consistently high values, although lower scores from older individuals should be particularly considered for further development. The well-rated SUS and the extensive use by users demonstrated that users could quickly learn to handle the system. Users also found the news topics and suggestions provided by the agent interesting.

As demonstrated by conversation examples, the agent could benefit from the graph structure. Users could thus query news about related entities of articles, as entities are directly stored with the articles. This allowed users to delve deeper into a topic, which showed that the approach of using a knowledge graph is beneficial. To delve deeper into a topic, the entity-based search was used, which was also generally the most frequently used feature. It is essential to provide the user with the relevant and appropriate news for the requested entity. A suitable match had to be found to correctly assign the user's statement to the right entity, which generally worked well. This was particularly the case for entity classes such as countries, persons, and cities. The system also proved effective in handling paraphrases and semantic matches. Thus, research question 3 is answered. As users most frequently used the

entity-based search and endorsed the navigation, our interaction patterns set out in research question 2 were generally confirmed. With the agent validated in the user study, based on the interaction patterns from research question 2 and the constructed knowledge graph from research question 3, it was shown that the problems identified in research question 1 were solved. The highly rated general usability of the system and its extensive use by users suggest that the developed conversational agent for German news search and exploration was successfully implemented, and both the basic and advanced functions of the system were accepted and utilized by the users.

6.2. Limitations

The conducted user study demonstrated the basic functionality of the system, yet also revealed several limitations. Firstly, delivering accurate news to the user about their searched entity is paramount, but the system occasionally faced difficulties, as outlined in Table 5.7.

For example, the agent had trouble recognizing acronyms unless they were spelled out. Inquiries about "AfD" returned incorrect results, while "Alternative für Deutschland" returned accurate news. This challenge could be resolved by expanding the graph with acronyms. Another issue occurred in the form of disambiguation: even if the entity was correctly identified, news was often returned in the wrong context. For instance, a user wanted news about the animal wolf, but received news about an investor with the last name Wolf. Speech recognition also had difficulties with words from another language. They were often misrecognized and led to a wrong entity assignment in the graph, even though the actual entity was present. For example, "ChatGPT" was recognized as "chaditip", which can be attributed to the Web Speech API being set to German. The threshold in fuzzy matching was intentionally set to a low value so that news is returned to the user more often. However, this resulted in a higher number of incorrect news being returned rather than informing the user that no news about the requested entity was available. For instance, instead of news about Justin Bieber, news about the animal beaver was returned. This could be resolved by setting a higher threshold or by employing a mechanism other than fuzzy matching. When a user requested news about multiple entities in combination, the response only ever concerned the first entity. For example, in news about China and Taiwan, only news about China was returned. This can be easily fixed by checking whether there are news items linked to both entities using the graph.

Another limitation was that some users encountered technical issues with the system, such as the microphone not working or the language being misrecognized. For this, the prototype presented in the study would need further refinement, perhaps through the development of an app or integration into a smartwatch. The choice of news sources was limited to Tagesschau, although additional news could be integrated into the graph, leading to more frequent correct responses, including regional news. An improvement in news quality could be achieved by summarizing entire articles, though accuracy must be maintained to prevent the spread of fake news. Simply adding more articles would lead to matches more frequently,

Table 6.1.: Entity recognition issues

Issue category	Description	Example dialogue excerpt
Acronyms	Entities were not recognized as acronyms, but they were when spelled out.	User: News about "AfD" (Political party). Agent: Here is news about "Wolgograd" (City)
Disambiguation	Entities were correctly identified, however, they were recognized in the wrong context.	User: News about "Wolf". (Animal) Agent: Here is news about "Wolf" (Person with surname Wolf)
Foreign language	In a foreign language, entities were incorrectly recognized by speech recognition.	User: News about "chaditip" (ChatGPT) Agent: Here is news about "Charité" (Hospital)
Fuzzy matching	Due to a too-low threshold in fuzzy matching, incorrect entities were recognized.	User: News about "Justin Bieber" (Person) Agent: Here is news about "Biber" (Animal)
Multiple entities	If two or more entities were mentioned simultaneously, only one entity was considered.	User: News about "China and Taiwan" Agent: Here is news about "China"

but this contradicts the agent's original design for current news. Another limitation arises from systems like the Wikifier, which is based on the knowledge base of Wikipedia and therefore cannot recognize brand-new events not yet contained in the knowledge base. A possible solution could be simple string matching with article headlines. While the entity-based news search was frequently used, category search was rarely employed. This could be due to the intent of category search often being misallocated. Another point of criticism was that the system was not perceived as particularly human-like. Some users complained that normal conversation in everyday language was not possible as the agent did not understand it. This could be improved by adding more training sentences from the newly acquired log data. Finally, the selection of news sources was limited, affecting the diversity and quality of the generated news. An expansion to regional and thematically diverse news sources could lead to a more comprehensive and accurately tailored news offer. In summary, the aforementioned limitations are important aspects that should be taken into account for the future improvement and development of the system.

7. Conclusion

7.1. Summary

This thesis investigated whether the use of knowledge graphs in the development of a conversational agent could improve German voice-based news search and exploration. Initially, existing systems were analyzed through literature review and comparative tests to identify their drawbacks related to news search. In this process, many fundamental problems were discovered and numerous opportunities for improvement were identified, such as the fact that aside from listening to podcasts, there are no real options for voice-based news search. Moreover, it became evident that there is a significant lack of research in this area. To find a solution to these still unresolved problems, a prototype was developed. The interaction patterns supported by the conversational agent were designed to address the problems identified in the comparative tests. The main focus was placed on entity-based news search, which allows the user to search for news on any topic. Additionally, the user was given options to control their news consumption or to get a brief daily overview.

Subsequently, a knowledge graph and a conversational agent were created to implement these identified interaction patterns. The knowledge graph consists of numerous articles from various categories to enable a comprehensive news search. The conversational agent was developed to be robust and user-friendly to provide the best possible experience for users. In a concluding two-week user study, we examined how understandable the concept was to participants and which interaction patterns the users ultimately used. It turned out that users often used entity-based news search to get more detailed information about the desired topic. The options for controlling news consumption were also frequently used. In addition, a questionnaire collected user experiences and evaluated the system's usability. It was found that users generally considered the system understandable and would like to use this system more frequently. It also demonstrated how users' exploration opportunities benefited from the structure of the knowledge graph, as users benefited from improved and more relevant recommendations. Thus, it was shown that the system can solve the problems identified at the beginning. These findings confirm the efficacy of the developed prototype using a knowledge graph and its interaction patterns in improving German voice-based news search and exploration.

7.2. Future work

Following the creation and implementation of the initial prototype, which received positive feedback, there are numerous opportunities for future work to further research in this field. To make the existing system more robust, it could be refined based on the results of user studies and tested over a longer period with participants. This would help to investigate how users would interact with the system over multiple interactions, not just initially. Integrating the system into a smartwatch or another existing system with a more user-friendly interface could facilitate this.

Another key aspect of future work involves significantly enhancing the capabilities of the system through the use of large language models. These could help the system understand a broader range of user requests and reduce erroneous matching between user requests and the responded news topic. As a result, the system could appear more natural and flexible. Additionally, large language models could enable the system to summarise news more concisely, thereby facilitating more personalized news briefings or quickly adapting news articles for voice-based news searches.

A further interesting step for research would be to explore how the system would function if used with data from a local newspaper. This could involve examining user behavior, particularly as listeners of a local newspaper currently have no access to voice-based news search since local news is rarely integrated into existing systems.

A. Appendix

Table A.1.: Evaluation form

Question type	**Question**
Evaluation questions on a 5-point Likert-scale	Please rate your overall satisfaction. Were the search results relevant to you? How easy was it for you to control news search? How interesting did you find the voice assistant's topic suggestions? Was the voice too fast or rather too slow for you? How did you find the length of the voice assistant's answers? How easy was it to get the voice assistant to understand what you wanted? How easy was it to understand the content of the voice assistant's answers? How understandable was the behavior of the voice assistant? How human was the interaction with the voice assistant?
Experience questions in a free-form text field	What are your experiences? List the advantages and disadvantages of the system. Have you not found any news on a topic? Do you have any suggestions for improvement? Do they have a desired function?
System Usability Scale questions on a 5-point Likert-scale	I think that I would like to use this system frequently. I found the system unnecessarily complex. I thought the system was easy to use. I think that I would need the support of a technical person to be able to use this system. I found the various functions in this system were well integrated. I thought there was too much inconsistency in this system. I would imagine that most people would learn to use this system very quickly. I found the system very cumbersome to use. I felt very confident using the system. I needed to learn a lot of things before I could get going with this system.

List of Figures

List of Tables

Acronyms

API application programming interface. 8, 21, 31, 33, 34, 36–39, 43, 44

CA conversational agent. 1–3, 8, 9, 13, 16–23, 41, 42, 44, 50, 52

DM dialogue manager. 4, 5

EL entity linking. 35–37, 39

JSON JavaScript Object Notation. 33, 34, 37, 39, 49

KG knowledge graph. 2–4, 7, 9–13, 16–21, 24, 29–35, 37–41, 47–50, 55, 56

NER named-entity recognition. 35–37, 39

NLP natural language processing. 1, 2, 4, 5, 11, 16, 18, 35

NLU natural language understanding. 4–6, 8, 41

RDF Resource Description Framework. 13–16, 66, 67

SSML Speech Synthesis Markup Language. 42, 48

SUS System Usability Scale. 56, 58–60

URI Uniform Resource Identifier. 13, 14

URL Uniform Resource Locator. 32, 36, 37

Bibliography

[1] G. Sinha, R. Shahi, and M. Shankar. "Human computer interaction". In: *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. IEEE. 2010, pp. 1–4.

[2] G. Terzopoulos and M. Satratzemi. "Voice assistants and smart speakers in everyday life and in education". In: *Informatics in Education* 19.3 (2020), pp. 473–490.

[3] Sara Lebow. *Voice assistant use is strong—revenue potential, not so much*. `https://www.insiderintelligence.com/content/voice-assistant-use-strong-revenue-potential-not-much`. Online; accessed 10 March 2023. 2022.

[4] C. Krettek. "ChatGPT". In: *Die Unfallchirurgie* 126.3 (2023), pp. 252–254.

[5] P. Schneider, T. Schopf, J. Vladika, M. Galkin, E. Simperl, and F. Matthes. "A Decade of Knowledge Graphs in Natural Language Processing: A Survey". In: *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online only: Association for Computational Linguistics, Nov. 2022, pp. 601–614. URL: `https://aclanthology.org/2022.aacl-main.46`.

[6] M. Allouch, A. Azaria, and R. Azoulay. "Conversational agents: Goals, technologies, vision and challenges". In: *Sensors* 21.24 (2021), p. 8448.

[7] K. K. Fitzpatrick, A. Darcy, and M. Vierhile. "Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (Woebot): a randomized controlled trial". In: *JMIR mental health* 4.2 (2017), e7785.

[8] A. Xu, Z. Liu, Y. Guo, V. Sinha, and R. Akkiraju. "A new chatbot for customer service on social media". In: *Proceedings of the 2017 CHI conference on human factors in computing systems*. 2017, pp. 3506–3510.

[9] Ottomatias Peura. *Advantages of Voice User Interfaces*. `https://www.speechly.com/blog/advantages-of-voice-user-interfaces`. Online; accessed 10 April 2023. 2021.

[10] H. Chen, X. Liu, D. Yin, and J. Tang. "A survey on dialogue systems: Recent advances and new frontiers". In: *Acm Sigkdd Explorations Newsletter* 19.2 (2017), pp. 25–35.

[11] A. Schmitt, T. Wambsganss, and J. M. Leimeister. "Conversational Agents for Information Retrieval in the Education Domain: A User-Centered Design Investigation". In: *Proceedings of the ACM on Human-Computer Interaction* 6.CSCW2 (2022), pp. 1–22.

[12] S. Vakulenko. "Knowledge-based conversational search". In: *arXiv preprint arXiv:1912.06859* (2019).

[13] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne. "Knowledge Graphs: Opportunities and Challenges". In: *Artificial Intelligence Review* (2023), pp. 1–32.

[14] B. Bui Huu Trung. *Multimodal Dialogue Management - State of the art*. Undefined. CTIT Technical Report Series 06-01. Imported from CTIT. Netherlands: Centre for Telematics and Information Technology (CTIT), Jan. 2006.

[15] M. F. McTear, Z. Callejas, and D. Griol. *The conversational interface*. Vol. 6. 94. Springer, 2016.

[16] J. Gao, M. Galley, and L. Li. *Neural approaches to conversational AI: Question answering, task-oriented dialogues and social chatbots*. Now Foundations and Trends, 2019.

[17] I. V. Serban, R. Lowe, P. Henderson, L. Charlin, and J. Pineau. "A survey of available corpora for building data-driven dialogue systems". In: *arXiv preprint arXiv:1512.05742* (2015).

[18] A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, E. King, K. Bland, A. Wartick, Y. Pan, H. Song, S. Jayadevan, G. Hwang, and A. Pettigrue. *Conversational AI: The Science Behind the Alexa Prize*. 2018. arXiv: `1801.03604 [cs.AI]`.

[19] H. Bunt. "Multifunctionality in dialogue". In: *Computer Speech & Language* 25.2 (2011), pp. 222–245.

[20] S. Young, M. Gašić, B. Thomson, and J. D. Williams. "Pomdp-based statistical spoken dialog systems: A review". In: *Proceedings of the IEEE* 101.5 (2013), pp. 1160–1179.

[21] D. Jurafsky and J. Martin. "Neural Networks and Neural Language Models". In: *Speech and Language Processing, 3rd Edition (Draft)* (2019), pp. 123–142.

[22] R. Lowe, N. Pow, I. V. Serban, L. Charlin, C.-W. Liu, and J. Pineau. "Training end-to-end dialogue systems with the ubuntu dialogue corpus". In: *Dialogue & Discourse* 8.1 (2017), pp. 31–65.

[23] J. Weizenbaum. "ELIZA—a computer program for the study of natural language communication between man and machine". In: *Communications of the ACM* 9.1 (1966), pp. 36–45.

[24] T. Winograd. "Understanding natural language". In: *Cognitive psychology* 3.1 (1972), pp. 1–191.

[25] S. J. Young. "Probabilistic methods in spoken–dialogue systems". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 358.1769 (2000), pp. 1389–1402.

[26] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[27] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[29] Z. Yan, N. Duan, P. Chen, M. Zhou, J. Zhou, and Z. Li. "Building task-oriented dialogue systems for online shopping". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[30] A. C. Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, D. Harter, T. R. G. Tutoring Research Group, and N. Person. "Using latent semantic analysis to evaluate the contributions of students in AutoTutor". In: *Interactive learning environments* 8.2 (2000), pp. 129–147.

[31] Z. Zhang, R. Takanobu, Q. Zhu, M. Huang, and X. Zhu. "Recent advances and challenges in task-oriented dialog systems". In: *Science China Technological Sciences* 63.10 (2020), pp. 2011–2027.

[32] S. Larsson and D. R. Traum. "Information state and dialogue management in the TRINDI dialogue move engine toolkit". In: *Natural language engineering* 6.3-4 (2000), pp. 323–340.

[33] A. Jacovi, O. B. El, O. Lavi, D. Boaz, D. Amid, I. Ronen, and A. Anaby-Tavor. "Improving Task-Oriented Dialogue Systems In Production with Conversation Logs." In: *Converse@ KDD*. 2020.

[34] S. Seneff. "TINA: A natural language system for spoken language applications". In: *Computational linguistics* 18.1 (1992), pp. 61–86.

[35] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. "PARADISE: A framework for evaluating spoken dialogue agents". In: *arXiv preprint cmp-lg/9704004* (1997).

[36] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. "Toward conversational human-computer interaction". In: *AI magazine* 22.4 (2001), pp. 27–27.

[37] D. Bohus and A. Rudnicky. "Error handling in the RavenClaw dialog management architecture". In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. 2005, pp. 225–232.

[38] M. Eskenazi, M. Strube, B. Di Eugenio, and J. D. Williams. "Proceedings of the SIGDIAL 2013 Conference". In: *Proceedings of the SIGDIAL 2013 Conference*. 2013.

[39] S. Oviatt. "Ten myths of multimodal interaction". In: *Communications of the ACM* 42.11 (1999), pp. 74–81.

[40] M. Walker, J. C. Fromer, and S. Narayanan. "Learning optimal dialogue strategies: A case study of a spoken dialogue agent for email". In: *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*. 1998.

[41] P. Dybala, M. Ptaszynski, R. Rzepka, and K. Araki. "Evaluating subjective aspects of HCI on an example of a non-task oriented conversational system". In: *International journal on artificial intelligence tools* 19.06 (2010), pp. 819–856.

[42] R. E. Banchs and H. Li. "IRIS: a chat-oriented dialogue system based on the vector space model". In: *Proceedings of the ACL 2012 System Demonstrations*. 2012, pp. 37–42.

[43] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. "A diversity-promoting objective function for neural conversation models". In: *arXiv preprint arXiv:1510.03055* (2015).

[44] L. Zhou, J. Gao, D. Li, and H.-Y. Shum. "The design and implementation of xiaoice, an empathetic social chatbot". In: *Computational Linguistics* 46.1 (2020), pp. 53–93.

[45] I. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. "Building end-to-end dialogue systems using generative hierarchical neural network models". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[46] O. Vinyals and Q. Le. "A neural conversational model". In: *arXiv preprint arXiv:1506.05869* (2015).

[47] M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley. "A knowledge-grounded neural conversation model". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[48] R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, and J. Pineau. "Towards an automatic turing test: Learning to evaluate dialogue responses". In: *arXiv preprint arXiv:1708.07149* (2017).

[49] A. Ghandeharioun, J. H. Shen, N. Jaques, C. Ferguson, N. Jones, A. Lapedriza, and R. Picard. "Approximating interactive human evaluation with self-play for open-domain dialog systems". In: *Advances in Neural Information Processing Systems* 32 (2019).

[50] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. "Deep reinforcement learning for dialogue generation". In: *arXiv preprint arXiv:1606.01541* (2016).

[51] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pp. 610–623.

[52] L. Fryer, D. coniam, R. Carpenter, and D. Lăpușneanu. "Bots for Language Learning Now: Current and Future Directions". In: *Language, Learning and Technology* 24 (June 2020), pp. 8–22.

[53] D. C. C. Bots, N. Sabharwal, and A. Agrawal. *Cognitive virtual assistants using Google Dialogflow*. Springer, 2020.

[54] L. Ehrlinger and W. Wöß. "Towards a definition of knowledge graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4 (2016), p. 2.

[55] H. Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic web* 8.3 (2017), pp. 489–508.

[56] I. Robinson, J. Webber, and E. Eifrem. *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

[57] T. Heath and C. Bizer. "Linked data: Evolving the web into a global data space". In: *Synthesis lectures on the semantic web: theory and technology* 1.1 (2011), pp. 1–136.

[58] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al. "Knowledge graphs". In: *ACM Computing Surveys (CSUR)* 54.4 (2021), pp. 1–37.

[59] Neo4j. *Movies Graph Example*. `https://github.com/neo4j-graph-examples/movies`. Online; accessed 5 March 2023. 2023.

[60] T. Wei, Y. Jiang, Y. Wang, Y. Luo, W. Lin, and Z. Chen. "Semi-automated construction of a knowledge graph with template". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 782. 3. IOP Publishing. 2020, p. 032054.

[61] S. Tiwari, F. N. Al-Aswadi, and D. Gaurav. "Recent trends in knowledge graphs: theory and practice". In: *Soft Computing* 25 (2021), pp. 8337–8355.

[62] A. Singhal et al. "Introducing the knowledge graph: things, not strings". In: *Official google blog* 5.16 (2012), p. 3.

[63] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. "Dbpedia: A nucleus for a web of open data". In: *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*. Springer. 2007, pp. 722–735.

[64] F. M. Suchanek, G. Kasneci, and G. Weikum. "Yago: a core of semantic knowledge". In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 697–706.

[65] D. Vrandečić and M. Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* 57.10 (2014), pp. 78–85.

[66] R. Angles, H. Thakkar, and D. Tomaszuk. "RDF and Property Graphs Interoperability: Status and Issues." In: *AMW* 2369 (2019).

[67] S. Das, J. Srinivasan, M. Perry, E. I. Chong, and J. Banerjee. "A Tale of Two Graphs: Property Graphs as RDF in Oracle." In: *EDBT*. 2014, pp. 762–773.

[68] R. Angles and C. Gutierrez. "Survey of graph database models". In: *ACM Computing Surveys (CSUR)* 40.1 (2008), pp. 1–39.

[69] M. A. Rodriguez and P. Neubauer. "The graph traversal pattern". In: *Graph data management: Techniques and applications*. IGI global, 2012, pp. 29–46.

[70] O. Lassila and R. R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Proposed Recommendation. World Wide Web Consortium, 1999. URL: `https://www.w3.org/TR/PR-rdf-syntax/Overview.html#basicSyntax`.

[71] F. Manola and E. Miller. "Rdf primer: W3c recommendation". In: *Decision Support Systems - DSS* (Jan. 2004).

[72] D. Allemang and J. Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[73] E. Prudhommeaux. "SPARQL query language for RDF". In: *http://www. w3. org/TR/rdf-sparql-query/* (2008).

[74] O. Ferrández, B. R. South, S. Shen, F. J. Friedlin, M. H. Samore, and S. M. Meystre. "BoB, a best-of-breed automated text de-identification system for VHA clinical documents". In: *Journal of the American Medical Informatics Association* 20.1 (2013), pp. 77–83.

[75] I. Cantador, A. Bellogın, and P. Castells. "A multilayer ontology-based hybrid recommendation model". In: *Ai Communications* 21.2-3 (2008), pp. 203–210.

[76] T. Young, E. Cambria, I. Chaturvedi, H. Zhou, S. Biswas, and M. Huang. "Augmenting end-to-end dialogue systems with commonsense knowledge". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[77] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26 (2013).

[78] B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, and L. Deng. "Towards end-to-end reinforcement learning of dialogue agents for information access". In: *arXiv preprint arXiv:1609.00777* (2016).

[79] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan. *Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset*. 2020. arXiv: 1909.05855 [cs.CL].

[80] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun. "Neural relation extraction with selective attention over instances". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 2124–2133.

[81] Z. Li, J. Kiseleva, and M. De Rijke. "Dialogue generation: From imitation learning to inverse reinforcement learning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 6722–6729.

[82] K. Zhou, W. X. Zhao, S. Bian, Y. Zhou, J.-R. Wen, and J. Yu. "Improving conversational recommender systems via knowledge graph based semantic fusion". In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020, pp. 1006–1014.

[83] L. Xu, Q. Zhou, K. Gong, X. Liang, J. Tang, and L. Lin. "End-to-end knowledge-routed relational dialogue system for automatic diagnosis". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 7346–7353. DOI: https://doi.org/10.1609/aaai.v33i01.33017346.

[84] Q. Chen, J. Lin, Y. Zhang, M. Ding, Y. Cen, H. Yang, and J. Tang. "Towards Knowledge-Based Recommender Dialog System". In: *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1803–1813. DOI: 10.18653/v1/D19-1189. URL: https://aclanthology.org/D19-1189.

[85] D. Chaudhuri, M. R. A. H. Rony, and J. Lehmann. "Grounding dialogue systems via knowledge graph aware decoding with pre-trained transformers". In: *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*. Springer. 2021, pp. 323–339. DOI: https://doi.org/10.1007/978-3-030-77385-4_19.

[86] H. Zhang, Z. Liu, C. Xiong, and Z. Liu. "Grounded Conversation Generation as Guided Traverses in Commonsense Knowledge Graphs". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 2031–2043. DOI: 10.18653/v1/2020.acl-main.184. URL: https://aclanthology.org/2020.acl-main.184.

[87] G. Wilcock and K. Jokinen. "Conversational AI and knowledge graphs for social robot interaction". In: *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2022, pp. 1090–1094. DOI: https://doi.org/10.1109/HRI53351.2022.9889583.

[88] N. Newman. *The future of voice and the implications for news*. 2018. URL: https://reutersinstitute.politics.ox.ac.uk/our-research/future-voice-and-implications-news.

[89] H. Sahijwani, J. I. Choi, and E. Agichtein. "Would You Like to Hear the News? Investigating Voice-Based Suggestions for Conversational News Recommendation". In: *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*. 2020, pp. 437–441. DOI: 10.1145/3343413.3378013.

[90] K. Yoshino and T. Kawahara. "News navigation system based on proactive dialogue strategy". In: *Natural Language Dialog Systems and Intelligent Assistants* (2015), pp. 15–25.

[91] P. Schneider, N. Rehtanz, K. Jokinen, and F. Matthes. "Voice-Based Conversational Agents and Knowledge Graphs for Improving News Search in Assisted Living". In: *arXiv preprint arXiv:2303.14286* (2023).

[92] D. Nadeau and S. Sekine. "A survey of named entity recognition and classification". In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26.

[93] W. Shen, J. Wang, and J. Han. "Entity linking with a knowledge base: Issues, techniques, and solutions". In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2014), pp. 443–460.

[94] Tomaz Bratanic. *Making Sense of News, the Knowledge Graph Way*. https://neo4j.com/developer-blog/making-sense-of-news-the-knowledge-graph-way/. Online; accessed 15 March 2023. 2021.

[95] J. Brooke. "SUS: A quick and dirty usability scale". In: *Usability Eval. Ind.* 189 (Nov. 1995).

[96] B. Rummel. "System Usability Scale (Translated into German)". In: (Apr. 2013). URL: https://www.researchgate.net/publication/272830038_System_Usability_Scale_Translated_into_German.