ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 3201
WSCG 2022 Proceedings

Chatbot Explorer: Towards an understanding of knowledge bases of chatbot systems

Alrik Hausdorf[1]
hausdorf@informatik.uni-leipzig.de

Lydia Müller[2,3]
lydia@informatik.uni-leipzig.de

Gerik Scheuermann[1]
scheuermann@informatik.uni-leipzig.de

Andreas Niekler[4]
aniekler@informatik.uni-leipzig.de

Daniel Wiegreffe[1]
daniel@informatik.uni-leipzig.de

[1]Image and Signal Processing Group, Leipzig University

[2]Institute for Applied Informatics (InfAI), Leipzig

[3]Leipzig University

[4] Computational Humanities, Leipzig University

ABSTRACT

A chatbot can automatically process a user's request, e.g. to provide a requested information. In doing so, the user starts a conversation with the chatbot and can specify the request by further inquiry. Due to the developments in the field of NLP in recent years, algorithmic text comprehension has been significantly improved. As a result, chatbots are increasingly used by companies and other institutions for various tasks such as order processes or service requests. Knowledge bases are often used to answer users queries, but these are usually curated manually in various text files, prone to errors. Visual methods can help the expert to identify common problems in the knowledge base and can provide an overview of the chatbot system. In this paper, we present Chatbot Explorer, a system to visually assist the expert to understand, explore, and manage a knowledge base of different chatbot systems. For this purpose, we provide a tree-based visualization of the knowledge base as an overview. For a detailed analysis, the expert can use appropriate visualizations to drill down the analysis to the level of individual elements of a specific story to identify problems within the knowledge base. We support the expert with automatic detection of possible problems, which can be visually highlighted. Additionally, the expert can also change the order of the queries to optimize the conversation lengths and it is possible to add new content. To develop our solution, we have conducted an iterative design process with domain experts and performed two user evaluations. The evaluations and the feedback from our domain experts have shown that our solution can significantly improve the maintainability of chatbot knowledge bases.

Keywords

application motivated visualization, chatbot, chatbot knowledge base, chatbot maintenance, decision tree.

1 INTRODUCTION

In recent years, advances in NLP (Natural Language Processing) have led to the development of more and more applications for text-based dialog systems, so-called chatbots. A user can interact with these programs using textual queries, and the chatbot's responses are then also delivered in text form, resulting in conversations with the chatbot. Possible use cases for chatbots include ordering processes, service requests, or the control of a smart home. Especially companies try to reach their customers by messenger with the help of a chatbot to reduce their costs. With the high request for systems that are easy to configure and cheap enough to be used by middle or small-sized companies, many tools were created. Chatbots are also used in many households to control the smart home, a well-known example being the voice-based Alexa system developed by Amazon.

Initially, the chatbot matches the user's input to a text item, the so-called *intent*, in its knowledge base. This part of the chatbot system is done by an internal NLP unit. Since these parts of the chatbot system are configured in a complex and language-specific way, most systems do not allow any modification of the configuration here. Next, the bot needs information about how to react to what is expected in response to the user's

request. Those reactions are called *actions* and can have very different functionalities.

The most common reaction of a chatbot is a text-based response. More complex actions could be to activate a light in the living room, select the pizza that the user wants to order, or register for an academic conference. If the bot cannot determine the intention of the user the bot can get the required information by further inquiries. For the pizza example, it is possible that the bot needs information about the cheese topping, which was not selected by the user beforehand. These conversations with the user are so called stories in the context of a chatbot. Stories are connected intents and actions that are configured by the maintainer to reach a specific goal.

Large companies like Google[1] or Microsoft[2] have complete systems to address all of these parts for their own business. However, there are many small companies with different requirements to make chatbots easier to run or configure. For example, the chatbot system s [3] tries to fill the gap of open-source software for this purpose.

However, there are currently still limitations in the use of chatbots. On the one hand, there are open issues in the area of natural language understanding to determine the user requests. On the other hand, it can be tedious to create a knowledge base for the chatbot. Both areas offer great potential for further improvements. In the current state, chatbot systems are mostly trained for one specific topic or task and new knowledge must be added manually. Furthermore, the maintenance tasks of chatbots are currently mostly done by domain experts and are barely assisted by visualizations. Thus, the use of chatbots is only possible productively with greater effort, since malfunctions in commercially used chatbots can lead to lost sales. However, small and medium-sized companies in particular cannot usually afford specialized employees for the support of chatbot systems. Therefore, these companies can only resolve problems or add new content with a great amount of time and effort.

In this paper we propose *Chatbot Explorer*, a system that enables users to get an overview of the chatbot knowledge base, to detect common errors and performance issues, as well as solve those issues with the assistance of visualizations.

Our contributions in this paper are:

- an interactive tree representation of knowledge bases for chatbots
- a semi-automatic detection of errors and problems in these knowledge bases
- interaction mechanisms for modifying and adding content to a chatbot
- a two-stage evaluation of the application using the System Usability Scale (SUS) standard

While creating the system the authors were in recurring contact with the domain experts. After developing the first version, an evaluation with 14 people was done to get valuable feedback on the system. That feedback was incorporated then to develop an improved version of *Chatbot Explorer*. This version was again evaluated with a larger group of participants to show the usability of the developed visualizations and interactions.

2 PROBLEM FORMULATION

For the setup of a chatbot, the maintainer must create a knowledge base using story elements. This mostly tedious task can lead to various errors so that the bot does not react correctly or unpredictably to user requests.

In cooperation with domain experts, we have identified the various sources of problems for this work and have developed mechanisms to detect and fix them. Our cooperation partner is a company that has developed and operated various chatbots for customers in the last years. At the beginning of the cooperation, we analyzed and tested their existing software systems. Additionally, we conducted interviews with the domain experts from their staff. During these interviews, the experts presented their tasks, demonstrated known problems during the setup of chatbots, and discussed possible improvements.

Together with the company a list of maintenance tasks, requirements, and problems that could occur, were identified. In the following, we describe the relevant tasks which are required to set up and maintain a chatbot:

T1 Add new knowledge to the chatbot or modify it to offer more information.

T2 Finding and fixing errors in the stories of a chatbot.

T3 Modification and reordering of the stories to optimize the conversations with the users so that they reach their desired objective faster.

Different chatbot systems mostly include methods to modify the knowledge base (T1). Besides those standard tasks, the experts are interested in a tool that helps with Tasks T2 and T3, too. In order to ensure that the tool can fulfill all three tasks, we have agreed with the domain experts on the following definition for a knowledge base for chatbots: A knowledge base can be reduced to the possible stories between the bot (actions) and a user (intents) [19]. Accordingly, a story is a goal-oriented conversation, for example, a user can order a specific product using the chatbot. A knowledge base of the chatbot then consists of several stories, for example, one

[1] https://cloud.google.com/dialogflow
[2] https://azure.microsoft.com/en-us/services/bot-services/
[3] https://rasa.com/

story for each product of the entire offer of a restaurant. Therefore, the maintenance of the knowledge base is equivalent to the management of the possible stories of the chatbot.

Following the structure of [8] the system requirements were collected together with the experts. In this process, we have identified additional requirements for the system, which are necessary for the successful processing of the tasks. First, the expert must gain an overview of the knowledge base of the chatbot in order to identify relations between the individual elements. On the other hand, knowledge bases can become very large, so the expert should be supported in identifying searched elements. This is especially important for the identification of structural errors, which should be detected (semi-) automatically.

As constraints for the visual system, the following were identified with the domain experts:

C1 The order in which the stories are stored in the knowledge base should not affect the visualization. Only the content of the stories should influence the visualization.

C2 Each story should be visualized as a continuous path.

3 RELATED WORK

Started in the 1960s by Joseph Weizenbaum [26], chatbots aim to entertain the user. One of the ambitious goals of chatbots is passing the Turing-test [24], by communicating like a real person. With the recent advances in the field of NLP, chatbot systems have gained a high amount of interest recently. Modern chatbots are used for instance as a replacement for FAQ systems or to enhance service for customers.

The wide range of possible applications for chatbots leads to a high amount of case studies. However, in this paper, we focus on surveys showing different aspects of chatbots. An overview of chatbot design techniques was created by Abdul-Kader et al. [1] in 2015 analyzing 9 selected studies. The survey by Chaves et al. [5] covers different interactions of bots. Maroengsit et al. [14] provides a survey about different evaluation methods of chatbots and provides a wide overview of chatbot systems. In 2019 Ch'ng et al. [6] give a summary of chatbots that are reported in the literature. Johannsen et al. [9] investigated 14 commercial chatbot providers in 2018 with a focus on supporting customer interaction. An overview of the training of a chatbot and how they perform on different models and methods of understanding the user is given by Csaky [7]. Klopfenstein et al. [12] give an overview of different conversational interfaces, patterns, and paradigms. These works highlight exemplary the state of the art for the deployment and usage of chatbots. However, they are not covering the area of developing and maintaining a knowledge base for a chatbot efficiently.

In the non-academic area, different systems analyze chatbot system data. The three most popular tools in this area are botanalytics [4], Virtual Agent Analytics by chatbase [5] and dashbot.io [6]. All three tools use the logs of chatbots and the collected metadata to provide insights into customer usage. All tools generate a tree structure for the analysis from the knowledge base and visualize the paths that users can follow in the conversations. However, these systems have been developed to show the use of chatbots and less to correct errors.

Several commercial tools are available to help the maintainer with creating and running chatbot systems. One of the largest open-source toolset for that purpose is RASA. With RASA it is also possible to visualize the chatbot knowledge base. For this purpose, the intents and actions are represented as nodes in a DAG (directed acyclic graph). This DAG is then represented using standard algorithms, but this is only suitable for smaller knowledge bases.
Recently, Yaeli and Zeltyn [27] presented a system to identify and fix problems and failures in chatbots. Their system can identify errors in productively deployed systems based on the conversation processes, so that the maintainer can then adjust the knowledge base. However, this requires a larger amount of conversational logs, which is why we chose a different approach for our system.

The usage of visualizations to gain insights into complex knowledge bases can help to understand data, communicate structures, and find the right decisions. Baumeister et al. [2] proposed a tree view-based visualization to show possible paths by creating nodes with different configurations. Renaud et al. [21] show why knowledge base visualization can help various target groups to understand the decisions of a system as well as a framework to increase the power of knowledge visualization. She highlights, the question "For whom?" has a high impact on the system. The work of Jonassen [10] shows that for problem-solving annotated directed graphs are efficient to assist the user with their already known mental representation of a problem. Neumann et al. [16] have used digital mind maps to structure knowledge for different audiences.

The visualization of conversational data is a common problem and was addressed by various works over time. Pascual-Cid et al. [18] developed a hierarchical tool to explore asynchronously online discussions using a hierarchical, radial layout. In the year 2012 Jyothi et al. [11] developed a similar visual tool to analyze asyn-

[4] https://botanalytics.co/
[5] https://chatbase.com/
[6] https://www.dashbot.io/

chronously students' interactions in online communications using radial, directed graphs. The work of Wattenberg et al. [25] uses a collapsible tree view to analyze a large number of conversations. The hierarchical representation of conversational data was proposed by Newman [17] using an icicle plot. All of these representations work with historically collected real-live conversational data. Her main goal is to make large conversations explorable and to give an overview of the conversation. The case of managing those stories or changing the structure of that conversations was not intended due to the used dataset.

None of the described approaches can fully cover all tasks and constraints as they are domain-specific. In the context of our work, we have therefore extended them to apply them to our problem setting.

4 DATA PROCESSING

Knowledge bases of chatbots can exist in different types as shown in Section 3. For *Chatbot Explorer*, we selected the stories as a representation of the data, since they are common for as many chatbot systems as possible.

The system works with the standard configuration files of RASA. This configuration consists of at least three files: domain.yml, story.md, and nlu.md. A specification of those parts of the files that are necessary for the system together with an example can be found in the supplemental material. The used configuration is, in general, the main source for a list of stories, intents, actions, and some meta-information like the messages that are used for the elements.

Chatbot Explorer was tested during the development with a self-created dataset that is supposed to represent a chatbot ordering a menu from a restaurant. We have generated this dataset from the offer of a large restaurant franchise. Therefore, each pizza and pasta was manually disassembled into their features. From the combination of those features, the first intent of each story was the type of the dish (intent "type pasta" or intent "type pizza"). Each story ends with an action, where the bot tells the user which menu s/he has ordered. The dataset contains 19 stories with 33 intents and 27 actions.

5 METHODS

Following the task analysis with the experts, we started with the development of visualizations that provide an overview of a given knowledge base. In the following, we investigated methods to interact with the visualizations and the underlying data. Afterwards, an automatic data analysis was conducted to highlight common problems in chatbot knowledge bases using structural information. These visualizations were created with the mantra of Shneiderman "Overview first, zoom and filter, then details-on-demand" [23] in mind.

Chatbot Explorer uses a tree visualization as an overview as well as a highly detailed view. The expert can apply filters, set highlights, and can zoom into the data. Details like intent examples are shown on demand. This tree representation was then adapted in the following process in order to improve the representation of the knowledge base.

5.1 Tree Layout

The action of a user of a chatbot in a conversation is a central point to decide what information is given to the user at the end. In general, the user requests the chatbot, whereupon the chatbot precipitates this request by asking questions to be able to execute the desired goal, for example ordering a product. These queries of the chatbot should thus be as precise and error-free as possible, so that the conversation is efficient. Therefore, the amount of decisions of the user is key to the performance of the chatbot. For that, we introduce the concept of story levels. If two conversations or parts of those conversations are on the same level, the number of decisions made by users is equal. Decisions where the user can not decide what answer s/he has to give, do not increase the level number. An example is when the chatbot asks for the user's customer number, the user cannot make their own decision for their answer.

Using a tree or graph for the representation of conversation paths is a common approach while using chatbots as already pointed out in Section 3. Existing approaches use an empty graph or tree and let the user manual fill with the desired data. However, our approach needs a suitable mapping from the knowledge base to the tree representation to work with an already defined knowledge base. This representation is created using a basic tree aggregation followed by several optional optimizations.

For the basic aggregation step, each story is integrated into the current tree. Each intent and action is represented as a node and each connection between them is stored as an edge. If a path in the tree already exists that is equal to a part of the added story, all shared nodes get only a reference to the newly added story. One common example is the greeting part of most stories, which merges into one node in the tree.

However, this approach can lead to unnecessarily sequential paths in the tree representation, since a given intent sometimes have only one possible action (e.g., asking for the name and address of the user). Therefore, we added a step optimization. The first approach is to combine intents and the following actions since this follows the idea that a question of the bot is a reaction to the text of the user. A second aggregation step combines in following all nodes that do not allow decisions of the user or the bot. Both aggregations decreased the drawing space of the tree significantly.

To improve the identification of the ending elements of a story, they are highlighted in the visualization. In addition, they can be separated from aggregated intent/action blocks by using a filter. Before the positions of the tree elements is calculated, the children of each node are ordered descending by the number of children. While the order only depends on the number of children after the aggregation, all stories are treated equally with respect to Constraint C1.

We improved the positioning of the nodes iteratively to the final, presented version. In the beginning, we used the default algorithm implemented in d3-tree. This algorithm uses the Reingold-Tilford [20] algorithm with an improvement of Buchheim et al. [4] to run in linear time. Since the nodes can differ in their height we switched to d3-flextree[7] that is size aware. However, the participants of the first evaluation stated, that the space-saving approaches of both algorithms sometimes make it unlikely to find a story end (leave). Figures of this first version can be found in the supplemental material as reference. Therefore, we decided to reimplement the layout on our own to achieve a better positioning of the nodes. This position algorithm starts to position all leaves on a horizontal line. Afterwards, the parent elements are positioned centered above all children. However, this positioning algorithm can create very large distances between children and parents, e.g., if individual children need a lot of height. To overcome this, we align the nodes afterward locally inside the level and/or globally regarding all children.

Each node (e.g., the element with the blue border in Figure 1 (2)) contains several elements that can be intents, actions, or the start element. A legend of the different elements is shown in Figure 1 (c). To distinguish the elements inside, the node are color-coded regarding their type. Additionally, each node is assigned to a level, which is indicated by the colored and labeled background area. The number on the top of each node represents the number of stories that share this path and the node. Below the edge of the node, the number of outgoing paths is shown.

Showing too much information at the same time can lead to visual clutter and reduce the readability of visualizations. Therefore, the visualization allows configuring which information should be shown for each element. Available information is provided as icons, representing the type of element (intent = user-icon, action = roboter-icon, start = play-icon), the name of the element, and a text example that could be used for that element.

5.2 Interaction

The following interactions with the tree view were identified during the requirement process or were suggested in the evaluations.

[7] https://github.com/klortho/d3-flextree

Pan and Zoom: In order to enable the user to explore the knowledge base it is possible to pan and zoom the view. If the zoom level reaches a certain level, the texts inside the elements are hidden and all boxes are transformed to squares with the same height as width. This "large-elements" drawing stretches the tree in the y-direction and helps to gain an overview even with larger trees on smaller screens.

Details on Demand: The expert may need more information about a specific element in the tree than is shown in the node. Therefore, we provide a tooltip mechanism that appears if the mouse hovers an element. To interact with the knowledge base the expert can select an element by clicking it (done so in Figure 1 (2)). *Chatbot Explorer* then offers the expert the following options: Modify the item, create a new story, and remove a story. Additionally, the highlighted element is shown at the bottom of the Treeview as a single element (see Figure 1 (2)).

Possible Problem Detection: Structural issues (see Task T2) are a serious problem that should be fixed before a knowledge base is used in a productive environment. Structural problems are often missed (see Section 2), causing problems after deployment. Therefore, *Chatbot Explorer* contains functionalities to detect and present these problems. The interface that summarizes possible problems within a given knowledge base is shown in Figure 1 (b). It shows the selected type, the currently selected problem, and how many problems were detected. The problem of "Undecidable paths" represents the error that after an intent of a user more than one possible reaction of the bot is possible. This is a problem since a non-deterministic answer can result in side effects, e.g., the bot has to decide, which way to follow. While selecting one of those "Undecidable paths" the parent-node, the children, and the paths between them are highlighted with a red border. That highlighting mechanism allows the user to use the overview to spot the position of the problem easily. The selected error can be found in Figure 1 (2). In the example, the intent "cheese_mozzarella" is followed by two actions: "utter_ask_topping_meat" and "utter_tell_pasta_cheese". If this problem occurs a chatbot engine randomly selects an action. Nevertheless this can lead to unexpected results. One possible solution to these errors can be the reordering of the queries the bot sends to the user or the addition of new queries to a story. To select a solution, however, a decision of the expert is necessary. Another common problem is the "Unnecessary step". It is defined as an action that asks something, followed by only one possible answer of the user (intent). For example, the bot asks the user for the cheese topping ("utter_ask_cheese") and the only answer that is possible is the selection of "cheese_mozzarella" (see Figure 1, bottom node on the left side in (2)). The bot system stays at that point of the story until the cor-

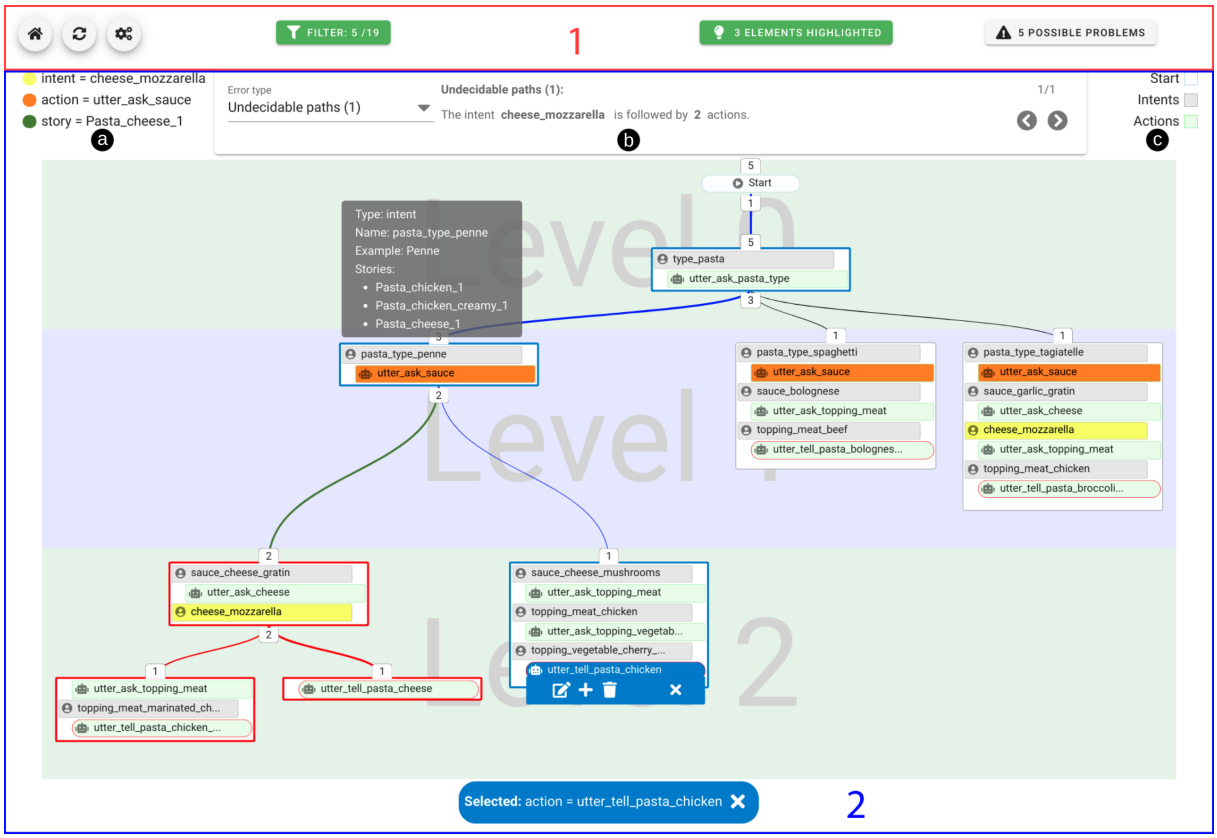

Figure 1: The main interface of the *Chatbot Explorer*. The interface can be separated into two basic areas: The configuration area (1) at the top of the view and the TreeView (2) below. Inside the Tree View (2) there is a legend of the used highlights (a), the possible problems interface (b), and color-scale of the node elements (c). Below that the zoomable and draggable tree of the selected stories are shown.

rect intent is provided by the user. The obvious solution is to remove this step from the story.

Filtering: To support the expert in the search for a particular element, *Chatbot Explorer* implements mechanisms to filter the stories by any intent, action, and story. While filtering for a story results in a single story, it is possible to invert every single filter and use that to remove specific stories from the analysis. Additionally, we implemented a user-defined highlight functionality to identify parts of interests faster and find relations between elements. One possible question regarding this is to identify elements where the bot asks questions about the sauce (see orange highlighted action "utter_ask_sauce" in Figure 1). While highlighting intents and actions shows the usage of those elements inside the tree, the highlighting of stories helps to find these stories in the tree more easily.

5.3 Manage Stories

In addition to the tree representation of the knowledge base, other visual representations were developed that experts can use to modify individual elements in the stories (see Task T1).

To create a new story the expert selects a starting point for the new content in the tree. For the new story, all parent nodes of the selected node are inherited to a new story, except the expert selects the root node. Since all stories consist of intents and action, *Chatbot Explorer*s shows all elements as a list, so the expert can add and remove them for the new story. To avoid ambiguities or missing aspects, the systems shows for each item in how many other stories this item is already used. For example, when modeling food orders, the expert can set up all processes as similar as possible for the customer. Since this task requires more than just rearranging existing elements, the expert can also create new intents and actions. Besides the creation of new stories, it is often necessary to modify existing stories. Therefore, the expert can select a node within the TreeView and activate the edit view. In this editing view (see Figure 2) the expert can select a story that uses this element. The selected story is centered in the view as a list. Additionally, a list of all available intents is shown on the left side of the view, and on the other side, a list of all available actions is shown. Equivalent to the creation of a new story, the story can be modified by dragging elements into the list, change the position of elements, or

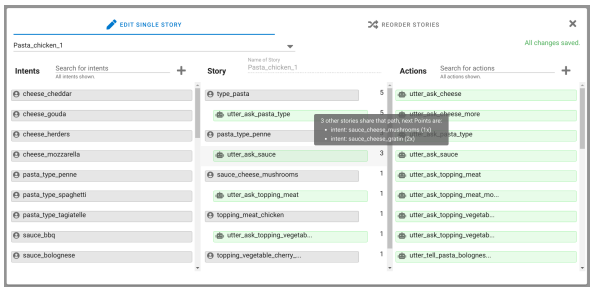

Figure 2: The edit stories interface of the *Chatbot Explorer* that is separated into three parts: The left part shows a list of all intents as well as a full-text search in the list and a button to add a new intent (plus-icon). The same structure is used on the right side for the actions. Between these two lists, the selected story is shown.

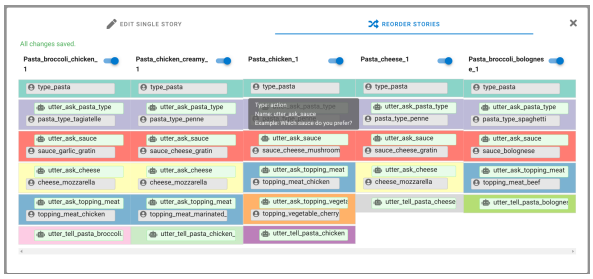

Figure 3: The reorder stories interface of the *Chatbot Explorer*. Each story can be disabled so that it does not get reordered along with the other stories. The elements were grouped by the type of question and the corresponding answer.

removing elements from the story. For some cases, the change of the knowledge base could require the deletion of a complete story. One example could be the change of the menu and a pizza can not be ordered anymore. For that purpose *Chatbot Explorer* allows the expert to select an element (see Figure 1 (2)) and use the delete button (trashbin-icon).

A further important aspect identified by the requirements process was the possibility of rearranging existing stories to make them more efficient according to certain criteria (see Task T3). For example, as a retailer, you want to be able to offer customers a short ordering process. For this purpose, *Chatbot Explorer* can identify nodes that are used by many stories. The expert can then activate the Reordering View (see Figure 3), where intents and actions of the stories are grouped. This is especially helpful if stories are very similar, for example in order processes. Therefore, we combine actions and the corresponding intents into groups within a story. Then these groups are compared between all stories using the action name and identical groups in the individual stories are colored with the same background color. The expert can then move blocks in one story and *Chatbot Explorer* automatically adjusts the position of identical blocks in all other stories.

5.4 System Architecture

To ensure that *Chatbot Explorer* is platform independent, we implemented a backend that provides a restful API to serve and import the knowledge base as well as changing it. The backend is written in JavaScript using NodeJs with an express-framework based API. The frontend is written in JavaScript using the VueJs framework for the interface. Some parts, like the color scales, were used from the d3 framework. For performance reasons, the visualizations used are a self-developed, CSS-styled, div-container structure. Due to its modular structure, *Chatbot Explorer* can also be easily integrated into existing systems. For this purpose, it needs access to the chatbot's knowledge base.

6 EVALUATION

While developing *Chatbot Explorer* we conducted two evaluations. Based on the interviews with domain experts, we developed the first prototype in close collaboration with them. This prototype was then tested by a user evaluation. In the first evaluation, several improvements were suggested, so we have extensively revised our visualization. After the implementation of the improvements was completed we conducted a re-evaluation of *Chatbot Explorer*. Both evaluations were using the questions of the SUS (**S**ystem **U**sability **S**cale).

The SUS was introduced by Brooke [3] in the year 1986 and has become since then a standard for evaluating the usability of a software system. It uses a 5-point Likert-type scale for 10 questions to calculate an overall usability score that is presented by a single value between 0 and 100. Following the recommendations of [13], we used the scale defined by Sauro and Lewis. To give the reader a better understanding of the calculated scores we added the grade-based interpretation of Sauro and Lewis [22] that maps the score values to the grades A to F. Additionally, we added a color-based interpretation by McLellan et al. [15] that uses green for excellent results (score between 85 and 100), yellow represents acceptable results (score between 65 and 84) and red represents all not acceptable results (below 65).

Since *Chatbot Explorer* was designed to help maintainers with limited knowledge about the usage of chatbots, there were no special requirements for the participants of the evaluations. Due to the COVID 19 pandemic, the participants had to perform the evaluation on their own hardware, but this also allowed us to test the accessibility as a result of the multiple hardware configurations. The devices used by the participants vary from laptops to workstations, FullHD to 4K displays, different amounts of available screens, and different systems (Linux, Mac, and Windows users). The used browsers were restricted to Chrome and Firefox. For the evaluation, we modified our test dataset to include various errors and problems that the participants were asked to identify and correct.

At the beginning of the evaluation, participants were asked to determine the number of intents that are necessary to order certain dishes. After that, the participants were asked to search for errors such as undecidable paths and unnecessary inquiries in the knowledge base of the chatbot. At the end of the evaluation, participants should rearrange the paths for certain stories in the chatbot to optimize the ordering process. Further descriptions of the use case can be found in the supplemental material. Each evaluation started with a prepared presentation to introduce *Chatbot Explorer* by the tester. For the second evaluation, we added for participants that did not participate in the first evaluation, two more slides for a more detailed view of the used dataset and how it was created. After the presentation, we provided a handout (see supplemental material) with three parts. The first part contains some support images to identify elements and problems. Afterward, multiple-choice questions were asked, which the participants had to answer. The last part contains more complex tasks, like changing the order of some bot queries. Additionally, all participants were asked to share their thoughts on the system while solving the tasks. After finishing the tasks, the tester had an open discussion with each participant about the use of *Chatbot Explorer*.

The form of the first evaluation consisted of the standard SUS questions and the following four additional questions:

1. The gender of the interviewed/participating person (male, female, other)
2. The age of the interviewed/participating person ($<$18, 18-29, 30-39, 40-49, 50-59, 60-69, $>$70)
3. The experience of the interviewed/participating person with chatbot-systems in general(5-point Likert-scale: 1 [no experience] ... 5 [expert])
4. If the interviewed/participating person has already worked with the tested system (yes, no, "I have seen pictures or a presentation of it")

For the second evaluation, we adjusted the questions for the experience level based on the findings of the first round (answer options: no experience, some experience, expert) and also ask whether the person participated in the first evaluation.

The first evaluation was attended by 14 persons (12 male and two female). Five persons had an academic natural language processing background, while seven persons worked at the visualization department. The other 2 participants were domain experts of our cooperation partner. In the second evaluation, all 14 participants from the first round attended again as well as six additional participants. Three of the 20 people were female and 17 were male. The background of the participants were: 5 people with a natural language processing background, 11 people with a visualization background, and 4 people from the associated chatbot company. The authors discussed the fact that the second evaluation could be biased by testing the same participants again. After collecting pro and cons we decided, that the reuse of those participants do not have a high impact since the visualizations and interactions comprehensively changed since the first evaluation.

The first evaluation showed some cases where our proposed visualizations did not perform satisfactorily. Also, some improvements were suggested by the participants. For example, many participants had problems with the proposed reordering mechanism due to a missing drag and drop mechanism. Another problem was that the test persons had difficulties in determining whether a branch of the tree represents a decision of the user. The participants also complained that they could only modify selected aspects of the knowledge base. Nevertheless, most participants appreciated the concept of the system and the possibilities of interaction. The first evaluation also shows that the person with more knowledge about chatbots sets a higher standard for the tool than people without experience. The granularity of the 5-point Likert-scale (values from 1 to 5) for the experience question results in one person for each of the values 2 and 4. To provide a better understanding of how the experience influences the results, the authors decided to reduce the experience scale of the first evaluation and aggregate them into three groups: no experience (value 1), some experience (value 2-4), and extensive experience (expert, value 5). The previous scale was mapped, aggregating values of 2 to 4 into the "some experience" group.

The mean of the SUS scores for all participants was a score of 77.9 (grade B+) and a median of 81.3 (grade A). The values according to the experience of the participants are shown in Table 1. Both values are acceptable but can be improved. Based on the results we decided that an additional round of implementing and improving the interaction mechanisms of the system would help to find a better solution.

In the second evaluation, participants who also participated in the first evaluation were more likely to indicate that they already had experience with chatbots in general. The values according to the experience level and if the participants were part of both evaluations can be found in Table 1. The mean of all scores in the second evaluation is 80.88 (grade A) and the median is 85 (grade A+). If one considers only the persons who already participated, the mean value is 79.82 (grade A-) and the median is 85 (grade A+). Those who participated for the second time also noted a significant improvement. All participants of the second evaluation recognized the usability of the *Chatbot Explorer* and expressed the opinion that the use of this system facilitates the maintenance of chatbot knowledge bases. The scores of each participant and

Type	First Evaluation				Second Evaluation					
	All	Experience			All	Participating		Experience		
		N	S	E		Y	N	N	S	E
Number of testers	14	7	6	1	20	14	6	9	7	2
Median	81.3/A	77.5/B+	85.0/A+	67.5/C	85.0/A+	85.0/A+	83.8/A	87.5/A+	90.0/A+	75.0/B
Mean	77.9/B+	75.4/B	82.5/A	67.5/C	80.9/A	79.8/A-	83.3/A	84.7/A+	80.0/A-	73.3/B-

Table 1: Results of the first and second evaluations. The results were obtained by using the following groups as columns: All (All participants that have done the evaluation), separated by the experience level and if they participated in the first evaluation. The columns of the "Experience" groups are: N=no experience, S=some experience, E=extensive experience. The columns of the "Participating" groups are: Y=participated in both evaluations, N=participated in second evaluation only. For each column, the amount of people in this group, as well as the median and the mean of the score-values of the participants in the group, is given.

for each question as well as the computed scores can be found in the supplemental.

7 DESIGN CHALLENGES

Several challenges were faced during the development of the final application that is described in this manuscript. One of the biggest challenges was to develop a visualization that could provide all the necessary information, but not overwhelm the user. This was especially important since Chatbot Explorer is explicitly also aimed at users who have no to little experience in creating and maintaining chatbots. Therefore, the first prototype focused on the visualization and analysis of the structure of the knowledge base. However, users had to identify common errors in knowledge based on the visualization. This turned out to be difficult and time-consuming, therefore we developed automatic error detection. The detected errors are presented in Chatbot Explorer and the user can analyze them individually. This has drastically reduced the required training time for new users.

Another challenge was to create an efficient and easy way of sorting many stories at once. The first version contains a list of stories to manually resort to each story. Based on the feedback from users, we added a function for editing more than one story in parallel is a useful addition as well as the user-friendly drag and drop possibility. The steps to the final described version of the "reorder stories" view contain many tested ideas like the traceability of changes and different types of grouping to assist the user.

8 CONCLUSION AND FUTURE WORK

Our work shows that visualization methods can help to gain a deeper understanding of the knowledge base of a chatbot. Chatbot Explorer can help the maintainer of a chatbot to identify different stories and to recognize parts of stories that s/he is interested in. The proposed tree visualization has the advantage of efficiently representing stories that share elements and the layer visualization helps to get a faster overview of the user's decisions. Additionally, the methods to rearrange the bot's queries and explore the resulting stories have proven to be useful to the experts.

We evaluated the system twice and got overall positive feedback. The participants and our domain experts emphasize the comprehensible visualization as well as the possibility to easily change all elements of a chatbot knowledge base using Chatbot Explorer.

In this work, we observed that changing the order of questions of the bot changes the appearance of several elements and the length of the paths. Therefore, it might be useful to develop a measurement of how certain changes affect corresponding stories and their efficiency to generate suggestions for possible changes and to be able to visualize the consequences of structural changes.

Acknowledgments

This research was funded in parts by the Development Bank of Saxony (SAB) under project numbers 100335729 and 100395769.

9 REFERENCES

[1] S. A. Abdul-Kader and J. Woods. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7), 2015.

[2] J. Baumeister and M. Freiberg. Knowledge visualization for evaluation tasks. *Knowledge and Information Systems*, 29(2):349–378, Nov 2011.

[3] J. Brooke. System usability scale (SUS): a quick-and-dirty method of system evaluation user information. *Reading, UK: Digital Equipment Co Ltd*, 43, 1986.

[4] C. Buchheim, M. Jünger, and S. Leipert. Improving walker's algorithm to run in linear time. In M. T. Goodrich and S. G. Kobourov, eds., *Graph Drawing*, pp. 344–353. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[5] A. P. Chaves and M. A. Gerosa. How Should My Chatbot Interact? A Survey on Social Characteristics in Human–Chatbot Interaction Design.

International Journal of Human–Computer Interaction, 0(0):1–30, 2020.

[6] S. I. Ch'ng, L. S. Yeong, and X. Ang. Preliminary Findings of using Chat-bots as a Course FAQ Tool. In *2019 IEEE Conference on e-Learning, e-Management e- Services (IC3e)*, pp. 1–5, 2019.

[7] R. Csaky. Deep Learning Based Chatbot Models. National Scientific Students' Associations Conference, 2019. https://tdk.bme.hu/VIK/DownloadPaper/asdad.

[8] M. Glinz. A risk-based, value-oriented approach to quality requirements. *IEEE Software*, 25(2):34–41, March 2008.

[9] F. Johannsen, S. Leist, D. Konadl, and M. Basche. Comparison of Commercial Chatbot solutions for Supporting Customer Interaction. In *ECIS 2018 Proceedings at AIS Electronic Library (AISeL)*, 2018.

[10] D. H. Jonassen. *Tools for Representing Problems and the Knowledge Required to Solve Them*, pp. 82–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[11] S. Jyothi, C. McAvinia, and J. Keating. A visualisation tool to aid exploration of students' interactions in asynchronous online communication. *Computers & Education*, 58(1):30–42, 2012.

[12] L. C. Klopfenstein, S. Delpriori, S. Malatini, and A. Bogliolo. The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, DIS '17, p. 555–565. Association for Computing Machinery, New York, NY, USA, 2017.

[13] J. R. Lewis and J. Sauro. Item Benchmarks for the System Usability Scale. *J. Usability Studies*, 13(3):158–167, May 2018.

[14] W. Maroengsit, T. Piyakulpinyo, K. Phonyiam, S. Pongnumkul, P. Chaovalit, and T. Theeramunkong. A Survey on Evaluation Methods for Chatbots. In *Proceedings of the 2019 7th International Conference on Information and Education Technology*, ICIET 2019, p. 111–119. Association for Computing Machinery, New York, NY, USA, 2019.

[15] S. McLellan, A. Muddimer, and S. C. Peres. The Effect of Experience on System Usability Scale Ratings. *J. Usability Studies*, 7(2):56–67, Feb. 2012.

[16] A. Neumann, W. Gräber, and S.-O. Tergan. *ParIS – Visualizing Ideas and Information in a Resource-Based Learning Scenario*, pp. 256–281. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[17] P. S. Newman. Exploring Discussion Lists: Steps and Directions. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, p. 126–134. Association for Computing Machinery, New York, NY, USA, 2002.

[18] V. Pascual-Cid and A. Kaltenbrunner. Exploring Asynchronous Online Discussions through Hierarchical Visualisation. In *2009 13th International Conference Information Visualisation*, pp. 191–196, 2009. doi: 10.1109/IV.2009.14

[19] A. M. Rahman, A. A. Mamun, and A. Islam. Programming challenges of chatbot: Current and future prospective. In *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 75–78, 2017. doi: 10.1109/R10-HTC.2017.8288910

[20] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, 1981.

[21] K. Renaud and J. van Biljon. A Framework to Maximise the Communicative Power of Knowledge Visualisations. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists 2019*, SAICSIT '19. Association for Computing Machinery, New York, NY, USA, 2019.

[22] J. Sauro and J. R. Lewis. When Designing Usability Questionnaires, Does It Hurt to Be Positive? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, p. 2215–2224. Association for Computing Machinery, New York, NY, USA, 2011.

[23] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pp. 336–343, 1996.

[24] A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 10 1950.

[25] M. Wattenberg and D. Millen. Conversation Thumbnails for Large-Scale Discussions. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, p. 742–743. Association for Computing Machinery, New York, NY, USA, 2003.

[26] J. Weizenbaum. ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine. *Commun. ACM*, 9(1):36–45, Jan. 1966.

[27] A. Yaeli and S. Zeltyn. Where and why is my bot failing? a visual analytics approach for investigating failures in chatbot conversation flows. In *2021 IEEE Visualization Conference (VIS)*, pp. 141–145, 2021. doi: 10.1109/VIS49827.2021.9623295