ALEX: AN ARTIFICIAL CONVERSATIONAL AGENT FOR STUDENTS AT THE TU BERLIN

Thilo Michael, Stefan Hillmann, Benjamin Weiss

Technische Universität Berlin, Quality and Usability Lab
thilo.michael@mailbox.tu-berlin.de

Abstract: User interfaces of most data-centered software systems are oriented on the data structure of the respective application. While such interfaces work well for trained users, they are most often not suitable for non-expert users. In this paper, we present ALEX, an advisory artificial conversational agent that uses machine learning and natural language processing to gather information from a relational database. Specifically, ALEX allows students at the Technische Universität Berlin (TU Berlin) to query courses and modules in a conversational way using textual input. The natural language understanding of the system is realized with a pipeline that uses a Hidden Markov Model tagger to annotate the input and groups phrases together to form SQL queries. ALEX was evaluated in a research study that compared the conversational agent with the existing online systems of the TU Berlin which are currently used by students to retrieve information about courses. The results show that the conversational approach is not only more efficient with non-expert users, but also has higher hedonic and pragmatic quality. A live demo of the tool can be found at `http://alex.qu.tu-berlin.de`.

1 Introduction

Most software systems that are currently used have interfaces that are heavily oriented on the data structure of the information they are providing. While this allows expert users to quickly pose complex queries to the database, this approach has the downside of being very hard to learn (in dependency to the complexity of the information stored in the database). While different user interface approaches like full-text search or query forms make the software easier to use, they limit the possibility to pose complex queries (i.e. joining multiple database tables).

In this paper we introduce ALEX, an advisory artificial conversational agent that serves students as a new interface for information about courses and modules at the Technische Universität Berlin[1]. We chose this area of application because courses and modules provide a wide range of data sets (e.g. teaching content of the courses, time and location of meetings, names of the lecturers) that can be queried coherently. ALEX, which was created as part of a master's thesis [1], allows for a user-driven conversational interaction, in which the user refines queries about the topic of interest. The queries of the user do not have to conform to a structure. As an example, the user might ask for courses of a specific professor by typing:

1. User: *"Which courses are given by Professor Dumbledore?"*
2. Alex: *"Here are all courses of Professor Dumbledore."*[2]
3. User: *"Show me only courses that are held on a Monday."*

[1] ALEX is designed to work in German. The examples in this Paper are translated into English.

[2] Displaying a list of courses. Compare Figure 1.

4. Alex: *"Here are all courses of Professor Dumbledore on a Monday."*[2]

This example shows that the system is able to continually refine the current query. In order to achieve this, ALEX keeps track of the progression of the conversation. The system also detects automatically, if a question from the user is meant to start a new query or not. It also asks counter questions, if more information for a successful evaluation is needed.

In this paper we outline the interface of ALEX and briefly describe technical details of the conversational agent. We also present a research study with 30 participants comparing the efficiency and usability of Alex and the existing system. In Section 2 we will provide context for the design and methods. In Section 3 we will describe the user interface, data structure and natural language pipeline of the agent and in Section 4 we will describe the setup and results of the research study that was carried out.

2 Related Work

ALEX is the continuation of an embodied conversational agent described in [2]. Since the introduction of human-like interaction with computer by Alan Turing [3], many different scopes for conversational agents have been defined [4]. While frame based agents have been developed for areas like booking systems [5] or the health sector [6], their application focus more on a machine driven conversation that conforms to a certain structure. This constraints the ways the user is able to ask questions. Goal oriented conversational agents, on the other hand, allow for a more natural conversation that still provides useful information on a certain topic [7] without requiring the user to have knowledge about the concrete data structure.

The approach shown in this paper combines the concepts of natural language based and goal oriented conversational agents to allow for a free goal oriented interaction while employing a natural language processing to be independent from the domain.

3 Natural Language Understanding in ALEX

ALEX is written in Python, and uses the `HiddenMarkovModelTagger` of NLTK[3]. The front-end is realized as a web application so that the complete system can be easily adapted for different applications or domains.

The graphical user interface is divided into two parts: a chat window that lets the user interact with the system and a result view that displays results in a structured way (see Figure 1). After sending a message to the system, an animation inside the conversation is indicating that the system is busy. The natural-language response of ALEX describes the query that was just executed and gives an implicit confirmation. The result view contains a table that lists entities that fit the users query. Depending on which information the user is asking for, the system displays different properties that might be useful.

The natural language pipeline of ALEX consists of four main steps. First, the input is sanitized and stop words are removed. Then, the Hidden Markov Model (HMM) tagger assigns labels to each word in the input sentence. The words and tags are then grouped into coherent phrases that represent a part of the query. These phrases are then converted into SQL statements and the query is executed. Finally, a natural language response is created on the basis of the query's result. The natural language response and the result are then shown to the user.

[3] See `http://www.nltk.org`

3.1 Preprocessing

To harmonize the processing of the users input, it is first tokenised and converted to lower case. After that, stop words are removed to enhance the performance of the tagging. Lastly, a list of specified keywords are replaced with their representative words. That way the diction of different users get converted to a single representation.

3.2 Hidden Markov Model Tagger

The HMM tagger is trained with a tagging schema that assigns to each word one of 6 different kinds of tags: R-tags, M-tags, C-tags, P-tags, Q-tags, and X-tags.

R-tags (short for *Return-tags*) are used for words that describe the kind of data the user is expecting and can be one of `R_LIST`, `R_SINGLE` or `R_COUNT`. For example, a user request starting with *"How many . . . "* would be tagged with `R_COUNT`, indicating that the result should be in the form of a number.

M-tags (*Model-tags*) are used for words that indicate which model the user is requesting. For example in the question *"Which course . . . "*, the word *course* would be annotated with `M_COURSE`, which indicates that the users is searching for a course, rather than a module or a person.

C-tags (*Constraint-tags*) are used for groups of words that constrain a field. For example in the question *"Show me courses on a Monday."*, *Monday* would be tagged with `C_CourseDate:day`. This indicates that the user wants to constrain the `day`-field of the `Course`-model to be *Monday*.

P-tags (*Property-tags*) are used for group of words that simply denote a property that should be included in the result. In the question *"Show me the type of examination of the module . . . "*, the *type of examination* would be tagged with `P_Module:typeOfExamination`. This indicates that the user wants to know the field `typeOfExamination` of the `Module`-model and that this information is included in the response.

Q-tags are used for words that modify a constraint and only occur alongside a constraint-phrase. For example, in the question *"Show me courses that begin after 8am."*, the word *after* would be tagged with `Q_GT` (greater than). This information is later combined with the constraint `C_CourseDate:startTime` to ensure that only course *after* 8am are displayed.

X-tags are used for words that do not add additional information to the query. While words tagged with an `X` might include useful information that is used at some point in the system (for example the word *"help"* would be tagged with `X_HELP`), words with X-tags are not considered in the query creation process.

As training data, we created a handmade list of 72 sentence-templates with the corresponding tags. These templates cover the common query types and some more complex queries. Before training, the templates were filled with information from the relational database of the existing course information system, resulting in over 500,000 sentences that were trained by the tagger.

3.3 Query Generation

With the input sentences and the resulting tags available, the system then proceeds to construct the corresponding SQL query. First, the system evaluates the M-tags found in the input sentence to determine the model (i.e. table) that should be queried. If none of those tags can be found in the input, the system tries to determine the model by analyzing the constraints given by the user.

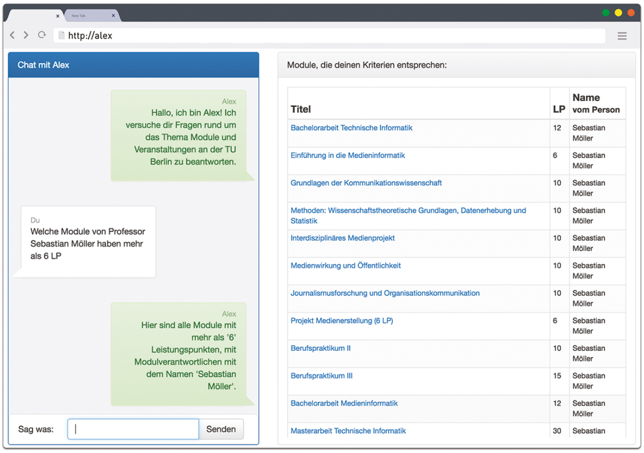
Then, the C-tags are clustered together with the Q-tags to form *filters*. Each filter builds a part of the WHERE-clause of the resulting SQL statement. A map that connects every model of the database through foreign keys is used to filter properties of other models than the one that is being queried. For instance, in order to find all Courses at a specific Institute, the system would create a path through the different models of the database that connects these two concepts. In this example, the filter would be "All Courses which are part of Modules that are given at a Chair that are part of the Institute X".

With the given filters, the target model and the history of previous requests, the system collects indicators reflecting if the current question addresses either a new query or an extension of the previous query. If a critical certainty is not reached, the systems prepares a counter question to ask the user if the previous query should be extended or not.

3.4 Response Generation

If a query can be executed and gives a non-empty result, the system creates a natural language response by combining phrases for each field constrained by the query. That way, the user is able to see how the system has interpreted the given input. Additionally to the description of the query, the system determines if the user asked for help, greeted or insulted the system and mixes this information into the natural language response.

For the list view of the query result (see Figure 1), the system selects a set of standard fields to display. These set of fields depend on the model that is being queried as well as what view is indicated by the Return-tag. Additional to these standard fields, the properties referenced by the Constraint- and Property-tags are added to the set of fields to be displayed.



Chat mit Alex

Alex
Hallo, ich bin Alex! Ich versuche dir Fragen rund um das Thema Module und Veranstaltungen an der TU Berlin zu beantworten.

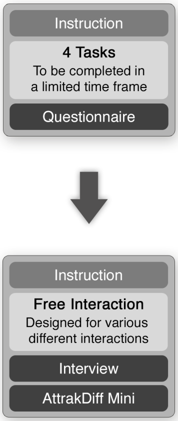
Du
Welche Module von Professor Sebastian Möller haben mehr als 6 LP

Alex
Hier sind alle Module mit mehr als '6' Leistungspunkten, mit Modulverantwortlichen mit dem Namen 'Sebastian Möller'.

Module, die deinen Kriterien entsprechen:

Titel	LP	Name vom Person
Bachelorarbeit Technische Informatik	12	Sebastian Möller
Einführung in die Medieninformatik	6	Sebastian Möller
Grundlagen der Kommunikationswissenschaft	10	Sebastian Möller
Methoden: Wissenschaftstheoretische Grundlagen, Datenerhebung und Statistik	10	Sebastian Möller
Interdisziplinäres Medienprojekt	10	Sebastian Möller
Medienwirkung und Öffentlichkeit	10	Sebastian Möller
Journalismusforschung und Organisationskommunikation	10	Sebastian Möller
Projekt Medienerstellung (6 LP)	6	Sebastian Möller
Berufspraktikum II	10	Sebastian Möller
Berufspraktikum III	15	Sebastian Möller
Bachelorarbeit Medieninformatik	12	Sebastian Möller
Masterarbeit Technische Informatik	30	Sebastian

Figure 1 – Screen capture of ALEX with the conversational area on the left and the result view on the right.



Instruction
4 Tasks
To be completed in a limited time frame
Questionnaire
Instruction
Free Interaction
Designed for various different interactions
Interview
AttrakDiff Mini

Figure 2 – Procedure of the research study.

4 Evaluation and Results

ALEX was evaluated in a research study with 30 participants and compared against a combination two online system currently used at the TU Berlin, namely MOSES[4] and LSF[5]. MOSES is a system used to query information about modules and contents of all majors at the TU Berlin. LSF is a system used to query information about courses and sessions that take place as part of modules.

Because a within-subject design could lead to a learning effect of the participants, a between-group design with two groups was chosen. The 15 participants who tested ALEX were between 21 and 35 years old (M = 27.5, 9 female). The other 15 participants who tested the baseline systems were between 20 and 34 years old (M=26.8, 7 female). All participants were students at German universities and familiar with the structure of majors into modules and courses. 12 of the 30 participants received financial compensation (10 €), while the other 18 were personally acquired volunteers.

4.1 Experimental Setup

Because a between-group design was chosen for the study, two version of the study were prepared. One half of the participants tested ALEX and the other half used MOSES and LSF. However, both tests consisted of the same tasks and questionnaires to keep the results comparable. The procedure of the study, which is depicted in Figure 2, is split in two main parts: a sequence of four tasks and a subsequent questionnaire about that interaction, followed by a free interaction with the system, an interview, and the AttrakDiff Mini [8] questionnaire.

In the beginning of the study, the participants were introduced to the system they were testing. The introduction consisted of a single page of text describing the tool(s) and a short live demo of the capabilities of the system(s).

After that, the participants worked on four predefined tasks that they had to complete in a time frame of 5 minutes per task. During the procedure, the systems were screen-captured and the audio was recorded. The first two tasks were rather simple and focused on one model (e.g. *"Find all modules in the mandatory study area of the bachelor computer science"*). The third and fourth task were more complex and required the system(s) to combine information from multiple models (e.g *"Find one course of the module 'lighting engineering' that is held on a Wednesday."*). For each task the degree of completion (completely solve, partially solved, not solved, given up), the number of interactions with the system, and the completion time were measured.

After completing the four tasks, the participants filled out a questionnaire about the interaction with the systems. The questionnaire consisted mainly of 9 sentences for each of which the participants had to mark if they agree or disagree on a 5-point Likert-scale. This questionnaire also contained an additional question for participants who tested MOSES/LSF that asked if the participants had used the systems before.

In the second part of the study, the participants were given a task relating to their own major, which led them to a longer interaction with the system(s). During the interaction, the computer screen and the audio was recorded. The test supervisor wrote down the main approaches of each participant.

After the free interaction, an interview was conducted of which the audio was recorded. The interview focused mostly on the overall experience of the tested system. Then, the participants rated the tested system with the AttrakDiff Mini questionnaire.

[4] Mathematisch Optimale Studenplan-Erstellungs-Software

[5] Lehre Studium Forschung

		ALEX		MOSES/LSF		
Task	**Feature**	**Avg**	± CI	**Avg**	± CI	**p-Value**
Task 1	Interactions	2.47	1.35	7.87	3.39	< 0.0001
	Time	75.67	37.29	178.67	78.90	0.0008
	Status	3	-	3	-	0.1939
Task 2	Interactions	3.00	2.22	6.47	3.85	0.0193
	Time	79.47	41.10	127.20	80.21	0.1030
	Status	3	-	3	-	0.0152
Task 3	Interactions	2.47	1.81	15.53	4.41	< 0.0001
	Time	60.27	24.10	295.87	53.18	< 0.0001
	Status	3	-	1	-	0.0037
Task 4	Interactions	2.47	1.35	12.53	3.48	< 0.0001
	Time	65.73	35.91	254.40	77.84	< 0.0001
	Status	3	-	1	-	0.0044

Table 1 – Comparison between the average and confidence (CL 95%) of Alex and MOSES/LSF for the number of interactions, time needed and status for each of the four tasks. The p-Values are calculated using Wilcoxon signed-rank test for the status and two-sided t-test with independent variables for the rest.

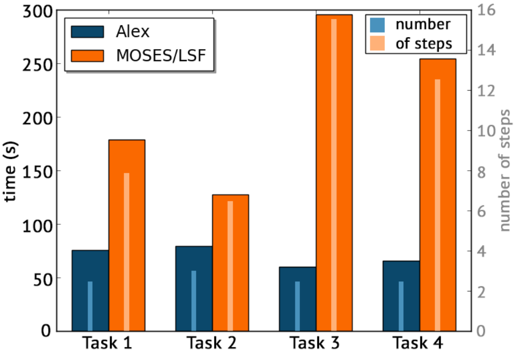


Figure 3 – Diagram of the average time and number of steps for the four tasks.

		ALEX		MOSES/LSF		
No.	**Statement**	**Avg**	± CI	**Avg**	± CI	**P-Value**
S1	I understood the system	3.40	0.44	2.13	0.44	< 0.0001
S2	The system understood my queries	3.33	0.43	2.67	0.43	0.0176
S3	The system was comprehensible	3.40	0.44	2.60	0.59	0.0178
S4	The system acted like I expected	3.00	0.39	1.80	0.33	< 0.0001
S5	The system was easy for me to query	3.27	0.42	2.20	0.51	0.0009
S6	The information was clear	3.47	0.44	2.47	0.59	0.0040
S7	The interaction was not too long	3.00	0.55	1.47	0.63	0.0002
S8	The interaction was fun	3.07	0.35	1.47	0.59	< 0.0001
S9	I would want to use the system again	3.53	0.31	1.73	0.57	< 0.0001

Table 2 – Average and confidence (CL 95%) for choices of each statement of the questionnaire. The p-values are calculated using the two-sided t-test with independent variables

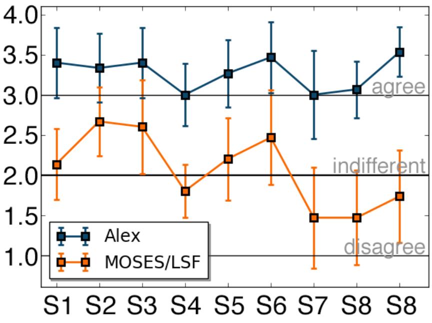


Figure 4 – Diagram of the average results and confidence (CL 95%) for each statement of the questionnaire. The statements (S1 through S9) can be found in Table 2

4.2 Results

The results for the first part of the study shows that for the baseline systems (MOSES and LSF, see Figure 3) the time for completion and the number of steps needed for the tasks is much lower for the first two simpler tasks than for the last two more complex tasks. Also, the median of the degree of completion of these systems went from *fully solved* to *not solved* for the last two tasks. In comparison, participants who tested ALEX took less time and number of steps for the tasks but also the results do not vary that much between the first two and last two tasks.

The analysis of the questionnaire handed to the participants after the four tasks shows that the perceived quality of ALEX is higher than that of MOSES and LSF. While participants who tested ALEX tend to *agree* with the sentences given, participants that tested MOSES/LSF are at average indifferent about the statements.

The results of the AttrakDiff Mini questionnaire that can be seen in Table 3 show that the ALEX outperforms MOSES and LSF in hedonic and pragmatic quality as well as in attractiveness.

The full study and results can be found in [1].

		ALEX		MOSES/LSF		
Group	Word pair	Avg	± CI	Avg	± CI	p-Value
Pragmatic Quality	complicated - simple	1.33	0.82	−0.93	0.41	< 0.0001
	impractical - practical	1.33	0.57	0.33	0.57	0.0090
	unpredictable - predictable	0.80	0.55	0.20	0.83	0.1843
	confusing - clearly structured	1.27	0.71	−1.40	0.43	< 0.0001
Hedonic Quality	tacky - stylish	0.33	0.69	−0.33	0.93	0.2020
	cheap - premium	1.40	0.58	0.27	0.92	0.0256
	unimaginative - creative	0.87	0.85	−1.13	0.73	0.0004
	dull - captivating	0.87	0.76	−0.60	0.69	0.0032
Attractiveness	ugly - attractive	1.53	0.49	−0.53	0.82	< 0.0001
	bad - good	2.00	0.49	0.40	0.62	< 0.0001

Table 3 – Comparison between the average and confidence (CL 95%) of the word pairs. p-Values (calculated using two-sided t-test with independent variables) higher than 0.05 are marked in red.

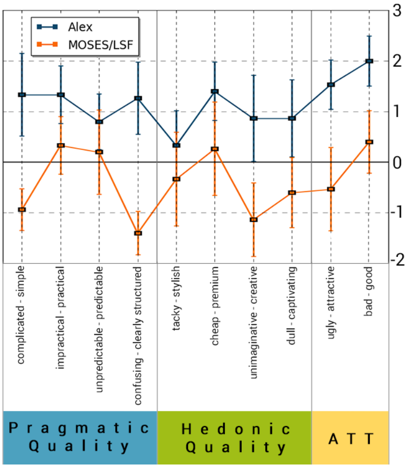


Figure 5 – Average and confidence (CL 95%) of the results for each word pair of the AttrakDiff Mini grouped into pragmatic quality, hedonic quality and attractiveness.

5 Discussion and Future Work

The results of the first part of the study shows that the conversational approach to retrieve information is more efficient and perceived to be more convenient by the users. Furthermore, the analysis of the AttrakDiff Mini shows that the user perceives the new approach as being more efficient and better to use. There are probably two main reasons for the described results. One reason being ALEX's natural language based user interface, which was easy to use by the participants. But also the fact that the querying process in MOSES and LSF is split into two different systems may play a role in these results.

The evaluation also revealed shortcomings of the conversational approach of ALEX. Problems named by the participants can be attributed to two kinds of shortcomings: technical problems, where the system did not behave as expected by the user, and conceptional problems, where users expressed that the conversational approach was not the ideal way to solve a task.

The technical problems mainly consisted of scenarios where ALEX was not able to retrieve the data the participant expected. The first reason for this kind of problem is that sometimes the system could not detect all necessary information in the input sentence to form a correct query. The second reason is that some types of questions were not envisioned when designing the system. While some of these questions asked for information that was not contained in the domain (e.g. sport courses) other questions were too complex for the system to resolve to a query (e.g. the question *"All [courses] that have similar teaching contents [to the courses in the last query]."*)

While some users used a short command-like interaction with the system, others tried to express themselves as clearly and unambiguous as possible. The latter behavior manifested itself in unnaturally long single sentences that contained information that was not necessary. While most often the system could answer those queries to a satisfying degree, the participants who posed those queries stated that the interaction took too long and was too complicated. Another form of conceptional problems arose with users who at first used the system with full sentences as intended, but later on shortened their inputs to use only necessary keywords. Those participants reported that once they understood the model through the output format they tried to adapt their queries to the data structure of the system. They stated in the interview that they like the natural language interface for basic querying but want to use a conventional interface once they were familiar with the underlying data model.

While the new approach shown in this paper shows a way to create a conversational agent that can be quickly remodeled for different domains, the evaluation shows that often times the precision of conventional interfaces are preferred for advanced users and also that some users have difficulties interacting in full sentences with a program. Future work may include the expansion of the training data, so that more parts of the domain can be queried in detail. Besides improving the accuracy and efficiency of the current approach, future work could also focusing on modifying the concept of ALEX. For example, a hybrid system could be conceptualized that combines the natural language input of a conversational agent with the search forms of conventional systems. This new approach could be designed in a way that lets users pose queries in a conversational way, but also lets them modify existing queries by changing filters through conventional input fields. This way users unfamiliar with the domain could interact with the system through the conversational agent and once they are more familiar with the domain, they could switch between using both approaches depending on the query they are trying to pose. Further research could also evaluate how spoken input and output changes the usability of the system.

References

[1] MICHAEL, T.: *Design and Implementation of an Advisory Artificial Conversational Agent*. Master's thesis, Quality and Usability Lab, Technische Universität Berlin, 2016.

[2] ENGELBRECHT, K.-P., B. WEISS, F. ADLER, B. LOUIS, A. MÜHLE, and E. SCHMITT: *Eca answering student queries for data collection and teaching*. In *2014 Spoken Language Technology Workshop. Symposium Proceedings*. IEEE, IEEE, 2014.

[3] TURING, A. M.: *Computing machinery and intelligence*. *Mind*, 59(236), pp. 433–460, 1950.

[4] O'SHEA, J., Z. BANDAR, and K. CROCKETT: *Systems engineering and conversational agents*. In *Intelligence-Based Systems Engineering*, pp. 201–232. Springer, 2011.

[5] MINKER, W., S. BENNACEF, and J.-L. GAUVAIN: *A stochastic case frame approach for natural language understanding*. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, vol. 2, pp. 1013–1016. IEEE, 1996.

[6] BICKMORE, T. and T. GIORGINO: *Health dialog systems for patients and consumers*. *Journal of biomedical informatics*, 39(5), pp. 556–571, 2006.

[7] CROCKETT, K., Z. BANDAR, J. O'SHEA, and D. MCLEAN: *Bullying and debt: Developing novel applications of dialogue systems*. *Knowledge and Reasoning in Practical Dialogue Systems (IJCAI)*, pp. 1–9, 2009.

[8] HASSENZAHL, M., M. BURMESTER, and F. KOLLER: *Attrakdiff: A questionnaire to measure perceived hedonic and pragmatic quality*. In *Mensch & Computer*, pp. 187–196. 2003.