Chatbot-based Tourist Recommendations Using Model-based Reasoning

Iulia Nica and **Oliver A. Tazl** and **Franz Wotawa**[1]

Abstract. Chatbots have gained increasing importance for research and practice with a lot of applications available today including Amazon's Alexa or Apple's Siri. In this paper, we present the underlying methods and technologies behind a Chatbot for e-tourism that allows people textually communicate with the purpose of booking hotels, planning trips, and asking for interesting sights worth being visit. In particular, we show how model-based reasoning can be used for enhancing user experience during a chat, e.g., in cases where too many possible selections are available or where user preferences are too restricted causing inconsistencies and as a consequence not possible answers to be provided. Besides the underlying foundations, we provide a use case from the intended tourism domain to show how such a model-based chatbot effectively can be used in practice.

1 Introduction

Communicating with systems based on natural language is very much appealing and of growing interest and importance also for industry. See for example [1, 2] for predictions about the rise of the chatbot market in the future. Natural language interfaces (NLI) offer a lot of new possibilities for humans to interact and collaborate with users [8]. Chatbots are a form of artificial intelligence system that allows a human-computer interaction in a natural language form. They could be based on rule sets or neural networks in order to decide the correct answer to the user's request. Chatbots are not restricted to certain application domains. They are flexible enough to be used in many different application scenarios and domains including systems for tourists recommending sights, hotels, or even complete travel plans. Often chatbots rely on pre-specified patterns that trigger the chatbot's behavior, e.g., see [25], restricting its space of interaction with users.

In this paper, we focus on improving adaptivity of chatbots in the context of recommender systems, where we have identified two issues that arise during and human-computer interaction session. In order to make a recommendation, the chatbot has to interact with the user in order to find out preferences and wishes in order to make an appropriate recommendation. In case of too little preference information, the *first issue* is, that a chatbot may *not be able to restrict the number of recommendations* to be provided to the user. Selecting a particular recommendation, e.g., the first one in a list of 1,000 elements might not be the best idea. It might also be not possible to find a general applicable function that returns the best solution for the current user. Hence, there is a need to further restrict the search space, which can be provided by asking the user about further preferences that allow to restrict the search space in an optimal way.

[1] TU Graz, Institute for Software Technology, Inffeldgasse 16b/2, A-8010 Graz, Austria, email: {inica, oliver.tazl, wotawa}@ist.tugraz.at

The *second issue* that may arise is the impossibility of providing even a single recommendation because of *inconsistent or too restrictive preferences provided by the user*. In this case, it is necessary, to provide feedback to the user and ask for removing preferences or for ranking preferences accordingly to their importance. In this paper, we focus on these two issues and provide a solution for both. For the *first issue*, we propose the use of Entropies for selecting new preferences. For the *second issue*, we suggest using model-based diagnosis for identifying the causes of inconsistencies. In addition, we put both parts together in a single recommendation algorithm for improving user experience when interacting with a chatbot.

The main contributions of this paper can be summarized as the follows.

1. An algorithm that is based on the ideas of model-based diagnosis and Shannon's information entropy to solve recommendation problems.
2. An iterative approach to the algorithm realized with a natural language interface using a chatbot.

The remainder of this paper is organized as follows: In the next section we introduce an example domain and give an overview of our algorithmic approach. Afterwards, we get into more details regarding the designed algorithms. Finally, we discuss related research and conclude the paper.

2 Chatbot for Tourism – A Case Study

In this section, we discuss the use of chatbots for recommendation in the area of tourism where we introduce a typical conversation between a tourist and a chatbot. This conversion presents a use case of a tourism chatbot and serves as motivating example throughout this paper. We depict the whole conversation in Figure 1, which illustrates the recommendation process for hotels in a specific area. There, after selecting the sight of interest, i.e., the Branderburg gate, the tourist requests a low-priced hotel near this sight. The answer of the chatbot from Line 14 in Figure 1 introduce a first challenging situation of every recommender, i.e., an unsuccessful user query. An intelligent recommender should be able to deal with such a situation and provide a list of items which fulfill as many requirements as possible, and ideally explain the cause of trouble. Therefore, the next step for the recommender should be to automatically identify the minimal set of inconsistent requirements provided by the user. Furthermore, a flexible, interactive query-based approach should allow the user to choose from the identified inconsistent requirements, and as a consequence enhancing the recommendation. The second corner situation, which is illustrated in Line 18, deals with the case where we have too many hotels matching the customer requirements. In this situation,

```
1  Bot: Hello John, how can I help you today?
2  John: I would like to plan a trip.
3  Bot: Great! Are you interested in a specific region
4         or would you like to plan the trip by activities?
5  John: I am interested in a specific region.
6  Bot: Please click on map to pick the point of interest
7         or type in the place that you are interested in.
8   (John clicks a point on the map near the city Berlin)
9  Bot: OK, I will search for relevant sights near Berlin.
10 (Shows a list of relevant sights in Berlin.)
11 John: Show me the Brandenburger Tor.
12 (The bot shows several information about the Brandenburger Tor.)
13 John: Looks nice! Find a low-priced hotel near this sight.
14 Bot: I am sorry, but there are no hotels of this category near
15      Brandenburger Tor. Which requirement is not that important
16      to you? Price or distance?
17 John: Distance.
18 Bot: OK! There are 9 hotels available for you. You should
19      incorporate another filter. How about parking? Will
20      you have a car?
21 John: No, I do not need a parking facility.
22 Bot: OK! Now there are 4 hotels left. Here is the list: Hotel 7,
23      Hotel 10, Hotel 15, Hotel 18.
24 John: Cool! Is Hotel 15 available from 19.08.2018 to 25.08.2018
25       for two people?
26 Bot: Yes! Should I book it?
27 John: Yes.
28 Bot: Hotel booked!
```

Figure 1: An Example Conversation

the system should be able to generate a proposal for the user, suggesting which hotel attribute or feature should be further constrained to narrow down the number of relevant hotels. Here the chatbot could randomly offer alternative features to choose from, or even better, it could optimize the searching process by making use of an algorithm to determine the attribute, which if constrained, leads to the largest information gain. Note that we rely on a pure knowledge-based approach, i.e., we assume that no other filtering like collaborative or content-based is available.

name	distance	category	parking	price
Hotel 1	med	5-stars	false	med
Hotel 2	med	5-stars	true	high
Hotel 3	med	3-stars	true	low
Hotel 4	long	3-stars	true	low
Hotel 5	med	5-stars	false	high
Hotel 6	short	4-stars	true	high
Hotel 7	long	2-stars	false	low
Hotel 8	med	5-stars	false	high
Hotel 9	long	3-stars	true	low
Hotel 10	long	2-stars	false	low
Hotel 11	short	4-stars	true	high
Hotel 12	short	3-stars	false	med
Hotel 13	med	5-stars	true	high
Hotel 14	short	3-stars	false	med
Hotel 15	long	3-stars	false	low
Hotel 16	long	3-stars	true	med
Hotel 17	med	3-stars	true	low
Hotel 18	med	2-stars	false	low
Hotel 19	long	2-stars	true	low
Hotel 20	long	4-stars	true	med

Table 1: Item set for $type = hotel$

In the following, we further discuss the details of how to overcome the considered issues that arise during the recommendation process, i.e., not being able to provide a recommendation due to inconsistencies, and not being able to reduce the number of selection given the current requirements. In this discussion we focus only on those parts that are important for the case study for the sake of clarity. Therefore, we further consider only the items of type $hotel$ to be part of the knowledge base, which we depict in Table 1. There we further assume that each $hotel$ possesses a simplified set of attributes, containing the $name$ of the hotel, its $price$, defining the price range (low, medium, or high), its $category$ with the domain {2-stars, 3-stars, 4-stars, 5-stars}, the availability of $parking$ space being either true or false, and $distance$ (short, medium, or long). Note that $distance$ is a special attribute, as it represents the distance to the starting point introduced by the user in the current recommendation session and thus it has to be recalculated on demand and not actually stored in the knowledge base.

It is easy to see that the requirements specified by the customer in Line 13 cannot be satisfied by the items from Table 1:

$$R_1 = \{r_1 : distance = short; r_2 : price = low\},$$

as there is no low-priced hotel within $distance = short$ in the given assortment. Hence, we are interested in identifying that minimal set of requirements that when changed, lead to a recommendation for the customer. In our example, the situation is simple. The recommendation system determines that either $r1$ or $r2$ have to be relaxed (in the sense that the chosen requirement will not be further taken into consideration when computing the recommendation). Still, in more complex scenarios, when the user query implies more requirements, the solution is not so straightforward. For instance, if the query was: " Find a low-priced, 4-stars hotel near this sight.", then we would have to deal with the following user requirements:

$$R_2 = \{r_1 : distance = short; r_2 : price = low; \\ r_3 : category = 4 - stars\}.$$

There choosing r_1 alone as inconsistent requirement would not solve the problem, because, as one can see in Table 1, we still have no items that satisfy both r_2 and r_3. Hence, in order to automatically identify the minimal set of inconsistent requirements, we would have to make use of logical reasoning methods that are able to determine causes for inconsistencies, e.g., consistency-based reasoning, where we have to describe the inconsistent requirements problem as a diagnosis problem. The idea is not new and state-of-the-art knowledge-based approaches like [6, 7, 14, 21] compute minimal sets of faulty requirements, which should be changed in order to find a solution. In this paper, we take the idea and transfer it to the domain of chatbots. In addition, we do not need to come up with conflicts for computing diagnoses but instead compute inconsistent requirements directly from the given formalized knowledge. Furthermore, the idea of personalized repairs is covered by asking the user directly which requirements he or she prefers. We discuss the recommendation algorithm in detail in Section 3.

In the following, we now discuss the other particular situation, where a recommender would have to deliver too many items matching the customer's requirements. In order to determine which attribute selection is the best one for accelerating the searching process, we suggest computing the entropy of the items' attributes at the

name	distance	category	parking	price
Hotel 3	med	3-stars	true	low
Hotel 4	long	3-stars	true	low
Hotel 7	long	2-stars	false	low
Hotel 9	long	3-stars	true	low
Hotel 10	long	2-stars	false	low
Hotel 15	long	3-stars	false	low
Hotel 17	med	3-stars	true	low
Hotel 18	med	2-stars	false	low
Hotel 19	long	2-stars	true	low

Table 2: Solution list for the user query $distance = \{med, long\}, price = low$

first place. Using entropies for finding the best next selection in order to accelerate the overall search process is not new. For example, De Kleer and Williams [3] introduced a measurement selection algorithm for obtaining the next best measurement in order to reduce the diagnosis search space. In our case, we have a similar situation and adapt using Entropies for our purpose. Further not that entropy is a measure commonly used in decision and information theory to quantify choice and uncertainty. For more details on Shannon's information entropy, we refer the interested reader to [24].

Let us consider our case study. In this example, the user query $R = \{distance = \{med, long\}; price = low\}$ leads to the solution list given in Table 2. There we have a set of 9 hotels with the attributes $distance, category, parking$, and $price$. In order to further reduce the number of hotels provided by the recommender, we have to identify the next attribute that should be further constraint by the user. In case of entropy used for selection, we have to compute the entropy for each attribute first. For more details on Shannon's information entropy we refer the reader to [24]. For computing the entropy of an attribute X, we make use of the following formula from [24] where $P(x_i)$ is the probability that attribute X takes value x_i:

$$H(X) = -\sum_i P(x_i) \log P(x_i) \tag{1}$$

Entropy has several interesting properties. Among them, as Shannon mentions in [24], $H = 0$ if and only if all the $P(x_i)$ but one are zero. Thus only when we are certain of the outcome does H vanish, otherwise H is positive. In the other extreme case, for a given n, H is a maximum and equal to $\log n$ when all the $P(x_i)$ are equal to $1/n$.

Let us now make use of entropies for selecting the next best attribute. Hence, we compute the attributes' entropies as follows:

$$\begin{aligned} H(distance) &= -P(med)\log P(med) - P(long)\log P(long) \\ &= -1/3\log(1/3) - 2/3\log(2/3) \\ &= 0.92 \end{aligned}$$

$$\begin{aligned} H(category) &= -P(3s)\log P(3s) - P(2s)\log P(2s) \\ &= -5/9\log(5/9) - 4/9\log(4/9) \\ &= 0.99 \end{aligned}$$

$$\begin{aligned} H(parking) &= -P(t)\log P(t) - P(f)\log P(f) \\ &= -5/9\log(5/9) - 4/9\log(4/9) \\ &= 0.99 \end{aligned}$$

$$\begin{aligned} H(price) &= -P(low)\log P(low) \\ &= 9/9\log(9/9) \\ &= 0 \end{aligned}$$

From these figures we see that we obtain the maximum entropy for the attributes $category$ and $parking$, whereas the minimum entropy is computed for $price$. In order to make the best choice, the recommendation system offers the attribute with the largest entropy value, i.e., $category$ or $parking$ in our case, to the user and asks him or her to further constrain it via selecting a certain attribute value. If the number of the remaining recommendations still exceeds a predefined maximum number of recommendations, the described solution reduction process based on entropy continues with the second best entropy attribute as already described above. In the next section, we describe an algorithm implementing this process in more detail and also integrate it within a whole recommendation process loop.

3 EntRecom Algorithm

Before stating our recommendation algorithm, which is based on a diagnosis algorithm that is close to ConDiag [20], and on a method that applies Shannon's information entropy [24] for the attributes, we introduce and discuss basic definitions. We first formalize the inconsistent requirements problem, by exploiting the concepts of Model-Based Diagnosis (MBD) [3, 22] and constraint solving [4].

The inconsistent requirements problem requires information on the item catalog (i.e., the knowledge-base of the recommendation system) and the current customer's requirements. Note that the knowledge-base of the recommender may be consistent with the customer's requirements (i.e., the customer's query) and an appropriate number of recommendations can be offered. In this case, the recommendation system shows the recommendations to the customer and no further algorithms have to be applied. Otherwise, if no solutions to the recommendation problem were found, then the minimal set of requirements, which determined the inconsistency with the knowledge base, have to be identified and consequently offered to the user as explanation for not finding any recommendation. The user can in this case adapt the requirement(s) (relax it/them). Here we borrow the idea from MBD and introduce abnormal modes for the given requirements, i.e., we use Ab predicates stating whether a requirement i is should be assumed valid ($\neg Ab_i$) or not (Ab_i) in a particular context. The Ab values for the requirements are set by the model-based diagnosis algorithm so that the assumptions together with the requirements and the knowledge-base are consistent. In the following, we define the inconsistent requirements problem and its solutions.

We start stating the inconsistent requirements problem:

Definition 1 (Inconsistent Requirements Problem) *Given a tuple (KB, REQ) where KB denotes the knowledge base of the recommender system, i.e., the item catalog, and REQ denotes the customer requirements. The Inconsistent Requirements Problem arises*

when KB together with REQ is inconsistent. In this case we are interested in identifying those requirements that are responsible for the inconsistency.

For our example introduced in Section 2, there is a knowledge base KB capturing the rows of the Table 1. This can be formalized as follows: $(name = Hotel\_1 \wedge distance = med \wedge category = 5 - stars \wedge parking = false \wedge price = med) \vee (name = Hotel\_2 \wedge distance = med \wedge category = 5 - stars \wedge parking = true \wedge price = high) \vee \ldots$. We have formalized knowledge stating equations, i.e., saying that $distance$ cannot be $short$ and med at the same time, i.e., $distance = short \wedge distance = med \rightarrow \bot$. In addition, there are two requirements $REQ = \{R1, R2\}$, and for each requirement a variable Ab_{R1}, Ab_{R2} stating whether the requirement should be considered or not. The requirements $R1$, and $R2$ themselves can be defined using the following logical representation $Ab_{R1} = 0 \rightarrow (distance = short)$, and $Ab_{R2} = 0 \rightarrow (price = low)$ respectively. Obviously, when assuming all Ab_{Ri} (for $i = 1, 2$) to be 0, we obtain an inconsistency because there is no hotel matching the requirements. Therefore, an explanation for such inconsistencies is needed.

A solution or explanation to the inconsistent requirements problem can be easily formalized using the analogy with the definition of diagnosis from Reiter [22]. We first introduce a modified representation of (KB, REQ) comprising (KB_D, REQ) where KB_D comprises KB together with rules of the form Ab_R for each requirement R in REQ. The solution to the Inconsistent Requirements Problem can now be defined using the modified representation as follows:

Definition 2 (Inconsistent Requirements) *Given a modified recommendation model (KB_D, REQ). A subset $\Gamma \subseteq REQ$ is a valid set of inconsistent requirements iff $KB_D \cup \{\neg Ab_R | R \in REQ \setminus \Gamma\} \cup \{Ab_R | R \in \Gamma\}$ is satisfiable.*

A set of inconsistent requirements Γ is minimal iff no other set of inconsistent requirements $\Gamma' \subset \Gamma$ exists. A set of inconsistent requirements Γ is minimal with respect to cardinality iff no other set of inconsistent requirements Γ' with $|\Gamma'| < |\Gamma|$ exists. From here on we assume minimal cardinality sets when using the term minimal sets.

For our example, inconsistent requirements are $\{R1\}$ and $\{R2\}$. In both cases there are hotels available and we do not obtain an inconsistency any more. In the following, we describe the algorithm for providing recommendations in the context of chatbots.

Algorithm 1 **EntRecom** takes a knowledge base, a set of customer requirements, and the maximum number of recommendations, and computes all recommendations. Algorithm 1 is an iterative algorithm that starts with deriving a constraint model CM from the knowledge base KB and the customer requirements REQ. Such a constraint representation captures the semantics of the provided knowledge base and requirements. Following the ideas presented in [20], we use a constraint solver both to directly compute the recommendations and to determine the inconsistent requirements. Still, in contrast to ConDiag, which guarantees to compute all the minimal diagnoses up to a predefined cardinality, we are interested here only in the minimal cardinality diagnosis, that in our case translates to the minimal set of inconsistent requirements.

Therefore, in Step 2, we check the consistency of our model by calling **CSolver**, a constraint solver taking the set of constraints CM

Algorithm 1 EntRecom(KB_D, REQ, n)

Input: A modified knowledge base KB_D, a set of customer requirements REQ and the maximum number of recommendations n
Output: All recommendations S

1: Generate the constraint model CM from KB_D and REQ
2: Call $\mathbf{CSolver}(CM)$ to check consistency and store the result in S
3: **if** $S = \emptyset$ **then**
4: Call $\mathbf{MI\_REQ}(CM, |REQ|)$ and store the inconsistent requirements in $IncReqs$
5: Call $\mathbf{askUser}(\mathbf{IncReqs})$ and store the answer in $AdaptedReqs$
6: $CM = KB \cup (REQ \setminus IncReqs \cup AdaptedReqs)$
7: **go to** Step 2
8: **end if**
9: **while** $|S| > n$ **do**
10: Call $\mathbf{GetBestEntrAttr}(A_S)$ and store the result in a
11: $A_S = A_S \setminus a$
12: Call $\mathbf{askUser}(\mathbf{a})$ and store the answer in v_a
13: $S = \mathcal{R}(S, v_a))$
14: **end while**
15: **return** S

and returning the set of recommendations S. If no recommendation was found (the empty set is returned), then we have to identify the minimal set of inconsistent requirements. For this purpose, we call algorithm 2 $\mathbf{MI\_REQ}(CM, |REQ|)$. Algorithm 2 starts with assuming one faulty requirement ($i = 1$) and continues to search, if necessary, up to the number of existing requirements. The constraint solver is this time called restricting the solutions to the specific cardinality i (see Line 2). In Line 3, the function $\mathcal{P}$ is assumed to map the output of the solver to a set of solutions. The termination criteria before reaching $|REQ|$ is given in Line 4, where a non-empty solution obtained from the satisfiability check is returned as result. In case no solution is found, the empty set is returned (Line 8).

Algorithm 2 MI_REQ$(CM, |REQ|)$

Input: A constraint model CM and the cardinality of the requirements set $|REQ|$
Output: Minimal set of inconsistent requirements

1: **for** $i = 1$ **to** $|REQ|$ **do**
2: $M = CM \cup \left\{ \sum_{j=0}^{|REQ|} ab_j = i \right\}$
3: $\Delta_S = \mathcal{P}(\mathbf{CSolver}(M))$
4: **if** $\Delta_S \neq \emptyset$ **then**
5: **return** Δ_S
6: **end if**
7: **end for**
8: **return** $\emptyset$

When being back into the **EntRecom** algorithm, we call the function $\mathbf{askUser}$ in order to adapt the inconsistent requirements ac-

cording to customer preferences. Afterward the constraint model is updated, by mapping the new adapted requirements, and the solver is called once again for checking consistency. In Step 9, the algorithm checks repeatedly if the cardinality of the computed recommendations is greater than the predefined maximum number of recommendations. Within this loop, we first determine the attribute with the best entropy, by calling function $\mathbf{GetBestEntrAttr}$ and store the result in a. Note that the entropy of each attribute is computed considering the values from the current set of solutions S. Next, we update the remaining set of attributes, then ask the user again about the preferred values of attribute a, and store the answer in v_a. In Step 13, function $\mathcal{R}$ keeps only the recommendations where attribute a takes values from v_a. Algorithm 1 obviously terminates, assuming that **CSolver** terminates.

Algorithm 3 $\mathbf{GetBestEntrAttr}(A_S)$

Input: The set of attributes A_S, containing the attributes and their domains accessible using the function dom.
Output: a_{res} the attribute with the highest entropy

1: $a_{res} = null, ent = -1$
2: **for** $a \in A_s$ **do**
3: $e = \sum_{x \in dom(a)} -P(x) \log P(x)$ compare Equation 1
4: **if** $ent < e$ **then**
5: $ent = e$
6: $a_{res} = a$
7: **end if**
8: **end for**
9: **return** a_{res}

Algorithm 3 **GetBestEntrAttr** determines the first attribute having the highest entropy. The algorithm uses the set A_S providing the the domain d_a for each attribute $a \in A_S$. **GetBestEntrAttr** iterates over the set of attributes. In every step it calculates the entropy for the current attribute a. If this attribute has a higher entropy than the entropies of the previously selected attributes, this value is stored in ent. In addition, the attribute itself is stored in a_{res}. After the end of the iteration cycle, the attribute with the highest entropy value stored in a_{res} is given back as result. Obviously, the algorithm terminates providing a finite set of attributes.

With the provide algorithms a chatbot for recommendations can be build that is able to deal with inconsistent requirements as well as missing requirements in a more or less straightforward way making use of previously invented algorithms. We are currently implementing the algorithms into a chatbot environment in order to provide a solid experimental platform for carrying out different case studies.

4 Related Work

The use of model-based reasoning and model-based diagnosis in particular in the field of recommender systems is not novel. Papers like [7, 14, 21] compute the minimal sets of faulty requirements, which should be changed in order to find a solution. There the authors compute the diagnosis for inconsistent requirements, relying on the existence of minimal conflict sets. In [7], an algorithm that calculates personalized repairs for inconsistent requirements is presented.The algorithm integrates concepts of MBD with ideas of collaborative problem solving, thus improving the quality of repairs in terms of prediction accuracy. [21] introduces the concept of representative explanations, which follow the idea of generating diversity in alternative diagnoses informally, constraints that occur in conflicts should as well be included in diagnoses presented to the user. Instead of computing all minimal conflicts within the user requirements in advance, [14] proposes to determine preferred conflicts "on demand" and use a general-purpose and fast conflict detection algorithm for this task.

Among the authors who integrate diagnosis and constraint solving more closely, we may mention [5] and later on [26, 27], who proposed a diagnosis algorithm for tree-structured models. Since all general constraint models can be converted into an equivalent tree-structured model using decomposition methods, e.g., hyper tree decomposition [10, 11], the approach is generally applicable. [28] provides more details regarding the coupling of decomposition methods and the diagnosis algorithms for tree-structured models. Further on [23] generalized the algorithms of [5] and [26]. In [18] the authors also propose the use of constraints for diagnosis where conflicts are used to drive the computation. In [9], which is maybe the earliest work that describes the use of constraints for diagnosis, the authors introduce the using constraints for computing conflicts under the correctness assumptions. For this purpose they developed the concept of constraint propagation. Despite of the fact that all of these algorithms use constraints for modeling, they mainly focus on the integration of constraint solving for conflict generation, which is different to our approach. For presenting recommendation tasks as constraint satisfaction problem, we refer to [15].

Human-chatbot communication is a broad field. It includes the technical aspect as well as psychological and human aspects. Papers like [12, 31] show several approaches of implementing chatbots in several domains. [31] shows an artificial intelligence natural language robot (A.L.I.C.E.), as an extension to ELIZA [32], which is based on an experiment by Alan M. Turing in 1950 [30]. This work describes how to create a robot personality using AIML, an artificial intelligence modelling language, to pretend intelligence and self-awareness. In [12] the authors demonstrate the usage of chatbots in the field of tracking food consumption. Sun et al. [29] introduced a conversational recommendation system based on unsupervised learning techniques. The bot was trained by successful order conversations between user and real human agents.

Papers like [8, 13, 16, 33] address the topics user acceptance and experience. In [33] a pre-study shows that users infer the authenticity of a chat agent by two different categories of cues: agent-related cues and conversational-related cues. To get an optimal conversational result the bot should provide a human-like interaction. Questions of conversational UX design raised by [8] and [19] demonstrate also the need to rethink user interaction at all. The topic of recommender systems with conversational interfaces is shown in [17], where an adaptive recommendation strategy was shown based on reinforcement learning methods.

5 Conclusions

In this paper, we introduced and discussed a recommendation algorithm based on the concepts of model-based diagnosis and Shannon's

information entropy. The algorithm is intended to be used in a chatbot environment for the tourism domain to handle the user responses via a textual user interface. We presented the challenges to solve common problems in the decision process of a tourist who communicates with such a chatbot. The identified challenges included the case of too many offerings that are presented to the user during the recommendation process and the case of too less offerings, which are caused by inconsistencies between the available knowledge of the chatbot and the given user requirements obtained during a conversation session.

In the proposed approach, we use model-based diagnosis to resolve the inconsistent requirements problem and Shannon's information entropy for solving the issue of too large amounts of offerings by presenting attributes and their values that can be chosen by the user in order to restrict the number of recommendations. Both solutions can be easily integrated within a chatbot environment guiding the chatbot application during the recommendation process.

We are currently implementing the presented algorithms including an integration with an existing chatbot environment dealing with tourism recommender systems. The algorithm is purposed to be used in several other industries and service domains as part of our future work. In the future, we will use this implementation for carrying out experiments and user studies with the objective to show that the approach can be effectively used in practical chatbot settings.

Acknowledgements

Research presented in this paper was carried out as part of the AS-IT-IC project that is co-financed by the Cooperation Programme Interreg V-A Slovenia-Austria 2014-2020, European Union, European Regional Development Fund.

REFERENCES

[1] Chatbot Market Size And Share Analysis, Industry Report, 2014 - 2025. https://www.grandviewresearch.com/industry-analysis/chatbot-market. Accessed: 2018-05-07.

[2] Gartner Top Strategic Predictions for 2018 and Beyond. https://www.gartner.com/smarterwithgartner/gartner-top-strategic-predictions-for-2018-and-beyond. Accessed: 2018-05-07.

[3] Johan de Kleer and Brian C. Williams, 'Diagnosing multiple faults', **32**(1), 97–130, (1987).

[4] Rina Dechter, *Constraint Processing*, Morgan Kaufmann, 2003.

[5] Yousri El Fattah and Rina Dechter, 'Diagnosing tree-decomposable circuits', in *Proceedings* 14^{th} *International Joint Conf. on Artificial Intelligence*, pp. 1742 – 1748, (1995).

[6] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumptner, 'Consistency-based diagnosis of configuration knowledge bases', **152**, 213–234, (02 2004).

[7] Alexander Felfernig, Gerhard Friedrich, Monika Schubert, Monika Mandl, Markus Mairitsch, and Erich Teppan. Plausible repairs for inconsistent requirements., 01 2009.

[8] Asbjørn Følstad and Petter Bae Brandtzæg, 'Chatbots and the new world of hci', *interactions*, **24**(4), 38–42, (June 2017).

[9] Hector Geffner and Judea Pearl, 'An Improved Constraint-Propagation Algorithm for Diagnosis', in *Proceedings* 10^{th} *International Joint Conf. on Artificial Intelligence*, pp. 1105–1111, (1987).

[10] Georg Gottlob, Nicola Leone, and Francesco Scarcello, 'Hypertree Decomposition and Tractable Queries', in *Proc. 18th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-99)*, pp. 21–32, Philadelphia, PA, (1999).

[11] Georg Gottlob, Nicola Leone, and Francesco Scarcello, 'A comparison of structural CSP decomposition methods', *Artificial Intelligence*, **124**(2), 243–282, (December 2000).

[12] B. Graf, M. Krüger, F. Müller, A. Ruhland, and A. Zech, 'Nombot - simplify food tracking', volume 30-November-2015, pp. 360–363. Association for Computing Machinery, (2015). cited By 3.

[13] Jennifer Hill, W. Randolph Ford, and Ingrid G. Farreras, 'Real conversations with artificial intelligence: A comparison between human-human online conversations and human-chatbot conversations', *Computers in Human Behavior*, **49**, 245 – 250, (2015).

[14] Dietmar Jannach. Finding preferred query relaxations in content-based recommenders, 04 2008.

[15] Dietmar Jannach, Markus Zanker, and Matthias Fuchs, 'Constraint-based recommendation in tourism: A multiperspective case study', *Journal of IT and Tourism*, **11**, 139–155, (2009).

[16] A. Khanna, M. Jain, T. Kumar, D. Singh, B. Pandey, and V. Jha, 'Anatomy and utilities of an artificial intelligence conversational entity', pp. 594–597. Institute of Electrical and Electronics Engineers Inc., (2016). cited By 0.

[17] Tariq Mahmood, Francesco Ricci, and Adriano Venturini, 'Learning adaptive recommendation strategies for online travel planning', *Information and Communication Technologies in Tourism 2009*, 149–160, (2009).

[18] Jakob Mauss and Martin Sachenbacher, 'Conflict-driven diagnosis using relational aggregations', in *Working Papers of the 10th International Workshop on Principles of Diagnosis (DX-99)*, Loch Awe, Scotland, (1999).

[19] R.J. Moore, R. Arar, G.-J. Ren, and M.H. Szymanski, 'Conversational ux design', volume Part F127655, pp. 492–497. Association for Computing Machinery, (2017). cited By 1.

[20] Iulia D. Nica and Franz Wotawa, 'ConDiag – Computing minimal diagnoses using a constraint solver', in *Proc. 23rd International Workshop on Principles of Diagnosis (DX)*, (2012).

[21] Barry O'Sullivan, Alexandre Papadopoulos, Boi Faltings, and Pearl Pu, 'Representative explanations for over-constrained problems', **1**, (07 2007).

[22] Raymond Reiter, 'A Theory of Diagnosis from First Principles', **32**(1), 57–95, (1987).

[23] Martin Sachenbacher and Brian C. Williams, 'Diagnosis as semiring-based constraint optimization', in *European Conference on Artificial Intelligence*, pp. 873–877, (2004).

[24] C. E. Shannon, 'A mathematical theory of communication', *Bell system technical journal*, **27**, (1948).

[25] B. Abu Shawar and E. Atwell, 'Using corpora in machine-learning chatbot systems', in *International Journal of Corpus Linguistics, vol. 10*, (2005).

[26] Markus Stumptner and Franz Wotawa, 'Diagnosing Tree-Structured Systems', in *Proceedings* 15^{th} *International Joint Conf. on Artificial Intelligence*, Nagoya, Japan, (1997).

[27] Markus Stumptner and Franz Wotawa, 'Diagnosing tree-structured systems', *Artificial Intelligence*, **127**(1), 1–29, (2001).

[28] Markus Stumptner and Franz Wotawa, 'Coupling CSP decomposition methods and diagnosis algorithms for tree-structured systems', in *Proceedings of the* 18^{th} *International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 388–393, Acapulco, Mexico, (2003).

[29] Y. Sun, Y. Zhang, Y. Chen, and R. Jin, 'Conversational recommendation system with unsupervised learning', pp. 397–398. Association for Computing Machinery, Inc, (2016). cited By 0.

[30] Alan M. Turing, *Computing Machinery and Intelligence*, 23–65, Springer Netherlands, Dordrecht, 2009.

[31] R.S. Wallace, *The anatomy of A.L.I.C.E.*, Springer Netherlands, 2009. cited By 53.

[32] J. Weizenbaum, 'Eliza-a computer program for the study of natural language communication between man and machine', *Communications of the ACM*, **9**(1), 36–45, (1966). cited By 1052.

[33] N.V. Wünderlich and S. Paluch, 'A nice and friendly chat with a bot: User perceptions of ai-based service agents'. Association for Information Systems, (2018).