Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 200 (2022) 602–610

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Industry 4.0 and Smart Manufacturing

A Bayesian based Intelligent Troubleshooting System

I Yung*, Federico D'ambrosio, Alissa Zaccaria, Fabio Floreani

Consorzio Intellimech, Via Stezzano 87, Bergamo 24126, Italy

Abstract

Troubleshooting systems can bring different benefits in assets management, particularly for service operations, facilitating the diagnosis of problems and faulty components identification. However, these systems are commonly based on rigid computation logic unable to handle uncertainties. In this work, a knowledge-based system exploiting the Bayesian theorem was developed and applied in a troubleshooting tool that relies on human-machine interaction. The required knowledge and the algorithm were analyzed and tested to ensure robustness and self-learning capabilities. Subsequently, the system was implemented in an industrial environment, specifically from a crane manufacturing company. The algorithm is robust to errors and provides the possibility of not answering some questions. However, the system performance is highly dependent on the questions, both in terms of quantity (adequate number compared to possible failures) and quality (effective to discriminate among failures). Indeed, this work shows how the system knowledge enhancement by introducing additional questions can significantly improve the troubleshooting performance. Future developments may involve user-friendliness enhancement and self-learning implementation to add and update questions over time.

© 2022 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 3rd International Conference on Industry 4.0 and Smart Manufacturing

Keywords: Knowledge based system; Bayesian theorem; troubleshooting system; human support; self-learning

1. Introduction

The potentialities of expert systems in assisting the human workforce have been investigated over the years in different fields. The first computer-based consultation system, Mycin, was proposed in the medical field. Mycin aimed to support physicians in bacterial infections diagnosis, allowing the selection of the most suitable therapy [1]. In the manufacturing field, computer-aided troubleshooting systems have been deemed suitable to overcome the scarcity of experienced troubleshooters [2].

Highly skilled troubleshooters are commonly trained on the job over a long period. Indeed, it is very challenging for companies to replace such a skilled workforce. Moreover, the evolution of machines complexity introduces additional challenges to the troubleshooting process. In this regard, several computer-based troubleshooting systems

* Corresponding author. Tel.: +39-035-069-0366.
E-mail address: i.yung@intellimech.it

1877-0509 © 2022 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 3rd International Conference on Industry 4.0 and Smart Manufacturing
10.1016/j.procs.2022.01.258

were developed based on the knowledge collected by experts specialized in the identification of machine failures (e.g., [3]). However, these systems are commonly based on rigid computation logic and unable to handle the uncertainties (different from what human is typically capable of).

Machine learning algorithms, which represent the basis of modern approaches to artificial intelligence, can be exploited to solve this problem. Indeed, these algorithms, based on statistical rules, can compensate for typical uncertainties and human errors. The objective of the present work is to investigate and exploit the possible algorithms behind a well-known artificial intelligence game, Akinator, for a robust troubleshooting system implementation, tolerant to mistakes and uncertain answers.

The game Akinator is able to guess a character, either a real or an imaginary person, after asking the player several closed-ended questions even in the presence of some wrong answers. Besides its robustness, other significant features of Akinator include the following:

- Available answers are limited: yes, no, don't know, probably, probably not. The last three possible answers might allow the system to handle uncertainties;
- The questions order is not randomly selected. The game mostly starts with general questions and ends with specific ones;
- There are several repetitive questions. Namely, two different questions obtaining the same information can exist (e.g., a question asking the character's gender is female and another question asking if the character's gender is male).

If the system is not able to identify the character, a list of possible characters is presented, allowing the user to select the one they are referring to. If the character is not present in the list, the user is asked to introduce a new one with a brief description. The character in Akinator refers to a person which the system should identify among all the possible candidates. The candidates can basically be any form of entity that the system is expected to identify. that can also be in the form of objects, numbers or failures in the case of troubleshooting.

Akinator presents advantageous features that could be exploited for troubleshooting purposes, such as simple answers, question order, learning capability and robustness to wrong answers. In this paper, the characteristics of several similar systems are first illustrated (section 2). Then, the most promising algorithm is selected and applied to a test dataset (section 3) before its validation on an industrial dataset (section 4). Finally, the simulation results are presented (section 5), followed by the conclusions and future developments (section 6).

2. Applicable expert systems

The troubleshooting systems under consideration involve two subsystems. In particular, one subsystem is aimed at updating the probability of the characters after each answer, while the second one manages the selection of the order of the questions. In the following, several algorithms which accomplish similar functionalities are presented.

The expert system in the medical field, Mycin, was developed to assist physicians in the diagnosis of bacterial infections and, hence, in the identification of the most suitable therapy for the patients. This system asks the doctor a long series of closed-ended and simple textual questions, and, based on the answers, it ranks the possible diseases from the most to least probable ones. From the user perspective, the functionality is close to the expected system. However, decision rules are the backbone of this system. As such, it cannot handle either wrong answers or automatic learning capability.

Renyi-Ulam is a mathematical game where a player tries to guess an unnamed object or a number thought by the other player by asking yes/no questions and considering that one or more of the given answers may be wrong. A rigorous analysis of the strategies proposed in the literature to find the correct answer in the most efficient way possible, meaning with the least number of questions, was outlined in [4]. Possible questions are straightforward for specific environments, such as guessing an integer, in which the selection of the questions is predictable. In several other cases, the questions are commonly related to the characteristics of objects, such as people's attributes in Akinator and symptoms of faults in troubleshooting.

A similar game called 20Q is a long-time case study in the information technology community [5]. One proposal to solve this game was based on a simple neural network [6]. More specifically, the system was based on simple

operations, namely, additions and subtractions. The algorithm identifies the character or object that has the highest value over a course of answered questions. Note that the selection of each question is similar to a binary search method, meaning that the next question is chosen to approximately subdivide the group of characters/objects with the highest value into two subgroups. The algorithm is robust and features an ongoing self-learning mechanism based on the result of the character/object identification.

The Bayes theorem was proposed to construct similar applications [7]. In this case, the Bayesian probability determines the likeliness of each character to be the correct solution. The robustness requirement is fulfilled since no character will be discarded in the case of wrong answers. Moreover, the order of the questions is selected to minimize the entropy. Given the robustness and the efficiency of the question selection process, this algorithm was considered in the present work to realize the expected troubleshooting system.

To summarize, the 20Q game and the Bayes theorem based system are the ones that meet the expected behaviour of the troubleshooting system. In particular, these systems offer the following features: candidates' evolution, question selection, robustness to wrong answers and learning capability. In this work, the Bayesian probability was chosen to represent the evolution of characters' probability over the answered questions. This method was selected for its clarity to control the process. Indeed, the sum of all the probabilities is equal to one. Contrarily, the verification is not straightforward in the neural network based strategy. Furthermore, the Bayesian approach is less "heavy" from a computational point of view, thus more suited for an industrial application. For the question selection, both entropy and binary search were deemed of interest to be investigated. Thus, both algorithms were first compared through a simulation on a test dataset.

3. Simulation

The selected algorithms are described in the following. In particular, the candidate probability evolution over the answered questions and the question selection strategies are outlined. Subsequently, the test dataset is presented. Then, the sequence of initialization, learning and conclusion of a game is illustrated. Finally, the simulation results are analysed.

3.1. Candidate probability evolution

The Bayes theorem computes a conditional probability of an event, i.e. the probability of an event given prior knowledge of conditions. Provided that a normalization step is carried out after every probability update, i.e. after every answered question, the probability of each character P after an n number of answered questions is simplified into:

$$P(C_j|\langle Q_1, A_1\rangle, \cdots, \langle Q_n, A_n\rangle) = \prod_{i=1}^{n} P(\langle Q_i, A_i\rangle|C_j) \tag{1}$$

where Q_i denotes the i^{th} question, A_i denotes the respective answer and C_j denotes the j^{th} character.

3.2. Question selection

Two different methods were considered for the selection of the next question. The first one is based on the generalized entropy [8]. The entropy of a distribution represents the level of uncertainty [9]. The entropy is high when the distribution shows high uncertainty, as for a uniform distribution. The next question is selected to minimize the entropy among the available questions considering all the possible answers:

$$Q^* = \underset{Q}{\operatorname{argmin}} \sum_{A} H(Q, A) P_A \tag{2}$$

where $H(Q, A)$ denotes the entropy of the predicted objects probability distribution, i.e. when question Q_{n+1} will be answered, based on the current objects probability (the probability of every candidate after n answered questions):

$$H(Q, A) = H[P(C|\langle Q_1, A_1\rangle, \cdots, \langle Q_n, A_n\rangle, \langle Q, A\rangle))] \tag{3}$$

and P_A denotes the total predicted probability of all the characters:

$$P_A = \sum_C P(\langle Q, A\rangle|C)P(C|\langle Q_1, A_1\rangle, \cdots, \langle Q_n, A_n\rangle) \tag{4}$$

The second method considered is based on binary search. In this case, the next question is the one that approximately divides the group with higher probability into two. In the case of two possible answers (yes/no), the next question is chosen by selecting the minimum difference between the total predicted probabilities with the possible next question being answered yes and no as follows:

$$Q^* = \underset{Q}{\operatorname{argmin}} \left(\sum_C P(C|\langle Q_1, A_1\rangle, \cdots, \langle Q_n, A_n\rangle, \langle Q, \langle yes\rangle) - \sum_C P(C|\langle Q_1, A_1\rangle, \cdots, \langle Q_n, A_n\rangle, \langle Q, \langle no\rangle) \right) \tag{5}$$

The generalized binary search considers the minimum difference of the sum from all the possible combination of answers.

3.3. Test dataset

Before dealing with the troubleshooting application, the algorithm was tested on a simple problem similar to Akinator. A small information matrix was created on public figures and their possible attributes, upon which the questions were developed (Table 1).

Table 1. Information matrix as knowledge base to the system.

Nb	Questions	A Barack Obama	B Homer Simpson	C George Clooney	D Madonna	E Donatella Versace
1	Was your character a president?	yes	no	no	no	no
2	Is your character a cartoon character?	no	yes	no	no	no
3	Is your character more than 50 years?	yes	don't know	yes	yes	yes
4	Is your character a Youtuber?	no	no	no	no	no
5	Is your character female?	no	no	no	yes	yes

The number of needed questions can be theoretically derived from the number of characters. This is feasible if the questions are qualitatively sufficient, i.e., three questions can distinguish at most eight characters if the questions can divide the characters attributes according to the binary combinations. In the chosen example, additional superfluous questions were introduced to test the selected algorithms, particularly the questions order selection. For example, question 4 is not able to distinguish any character since the answer is the same for all of them. This question is added as the industrial matrix information is expected to hardly guarantee the availability of divisive questions at every question selection step.

3.4. Initialization, learning and ending a game

Before executing the simulation, two initial probabilities must be set: one assigned to each character at the beginning of each game, and a set of probabilities that associates the questions to the characters. The initial probability for each character is set uniformly, i.e., setting the probability of each character to 0.2. The initial probability of each answer can also be set equally when no knowledge is available, e.g. the probability of whether George Clooney has a dog is 0.5 yes and 0.5 no. This probability is converted to statistics count for learning purpose. For example, assuming 2 games have been played as the initialization, 1 yes answer and 1 no answer are the initial statistics count for the question of whether George Clooney has a dog. If initialized with 10 games, 5 yes and 5 no are the initial count based on the defined probability. Starting from an unknown condition for the association of all the questions to all the characters, the system is not able to guess any character correctly. Thus, the users have to perform the initial learning phase to update the information matrix by conducting a sufficient number of games in which the users must select the right character after answering all the questions.

In our example, the initialisation of the system can be based on a priori knowledge of the answers given the popularity of the characters. Thus, the features of each character, meaning the answers to the questions, are known. In this context, the learning phase can be omitted, and each answer can be converted to the probabilities shown in Table 2.

Table 2. Conversion to probability.

Answer	Assigned probability
yes	yes:0.9; no:0.1
no	yes:0.1; no:0.9

Based on this conversion, the character probability is updated during a game. For example, the probability of Barack Obama being thought by the player in a game is updated by 0.1 when the player answers “no” to question 1 “Was your character a president?” (Table 1). The information matrix filled with the probabilities associated with each character is the knowledge base of the system. The test results described below are based on these considerations.

3.5. Test results

This preliminary simulation was performed to understand the behaviour of the developed system before its application in the industrial case study. To compare the question selection algorithms and the kind of required questions, the game was simulated for different characters, summing up to 5 games in total for each question selection algorithm. The system ends when the probability of one character is higher than 0.5. This threshold is a tuning parameter that should be defined for each case. The simulation was based on both entropy and binary search as question selection strategies and the Bayesian system as character’s probability evolution. Table 3 presents the results in terms of questions order and the final probability of the selected character. The simulation of a game that has to guess Obama required 1 question based on entropy and 2 questions based on the binary search to reach the probability threshold and correctly guess the character, provided that all the given answers are correct. Similarly, Homer and Clooney are correctly guessed after two and three questions based on entropy question selection while three questions were asked based on binary search. The simulations concerning Madonna and Versace never reach the probability threshold to end the game due to a structural issue in the matrix information. Indeed, these characters have the same response to all the available questions (column D and E in Table 1).

Regarding the question order, the entropy based algorithm always poses question 1 in the beginning of every session.

Comparing the questions being asked to successfully guess the first three characters, the entropy based simulation requires less questions on average than the binary search based simulation. In addition, initialized with the same probability 0.2 for every character at the beginning of every session, the first question that minimizes the entropy of the predicted probability is question 4. This behavior is not preferred as question 4 is a completely unnecessary question (no is the answer for every character). In contrast, the simulation based on binary search asks question 4

Table 3. Simulation results.

Character	Entropy		Binary search	
	Order	Probability	Order	Probability
Obama	1	0.82	5,1	0.89
Homer	1,2	0.85	5,1,2	0.93
Clooney	1,2,5	0.82	5,1,2	0.82
Madonna	1,2,5,3,4	0.49	5,1,2,3,4	0.49
Versace	1,2,5,3,4	0.49	5,1,2,3,4	0.49

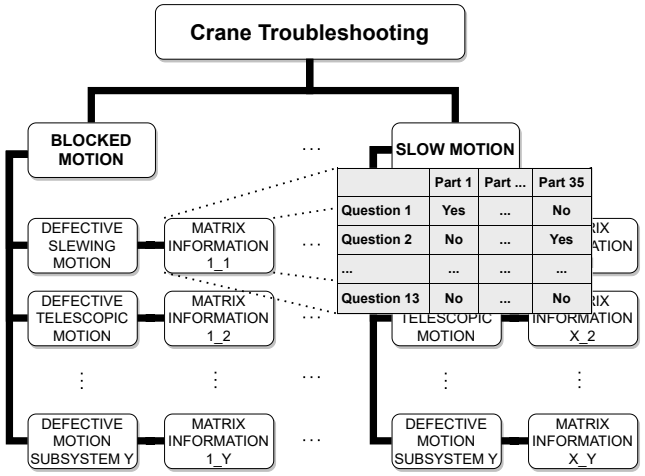

Fig. 1. The complete matrix information of a crane troubleshooting system

only to complete the series of questions if it is not possible to guess the character, that is the desired behaviour for our system. These findings lead to the use of binary search as a question selection strategy for the developed system.

Moreover, a robustness test was performed on the selected algorithm. The first evidence is that additional questions are needed to make the algorithm robust to possible wrong answers. A simple way to overcome this issue for example is to add a redundant question; in this case, it could be whether the character is male with opposite answers to question 5 for each character. Simulating the system with this trick, it is possible to guess Obama with 81% probability by correctly answering questions 6 and 1 and incorrectly answering question 5. The use of an additional question strengthens the system at the expense of a higher number of questions. Therefore adding questions unconditionally is not a good solution.Thus, it is necessary to properly design the system taking into account the right trade-off between robustness and usability.

4. Industrial case study

The developed methodology was finally applied to an industrial case. The identified industrial application is related to a manufacturing company that produces and sells hydraulic cranes. However, the algorithm could be adapted to other different industrial contexts.

The company currently uses a basic troubleshooting tool to support the operators in the identification of faulty components and related failure modes. This system is based on a decision tree model which, following the sequence of “yes” or “no” answers of the operator, reaches the target component and its failure mode. This structure has different inefficiencies:

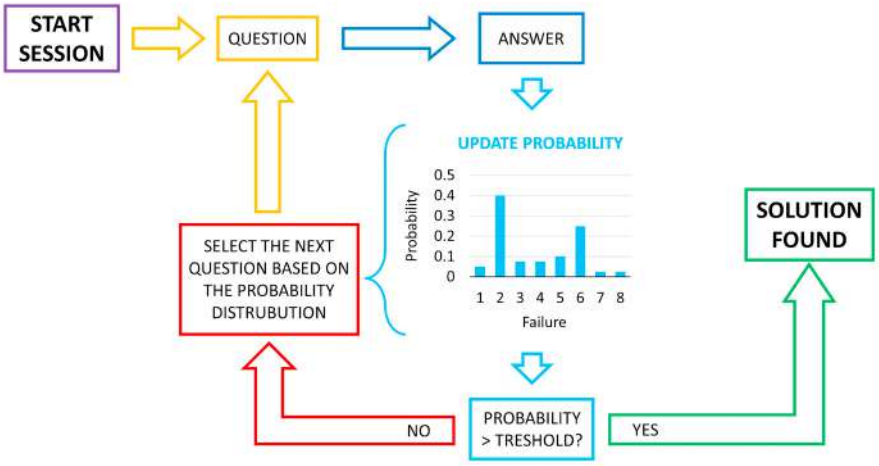

Fig. 2. Troubleshooting system workflow.

- the travel time of the tree is often excessive, even for simple failures which could be identified with a lower number of questions;
- the structure is rigid and the navigation gets stuck when the operator is not able to answer in a precise way to a specific question or is in a dubious situation;
- If the user provides a wrong answer, it can be difficult or even impossible for the system to reach the correct solution.

This work aims to improve the actual tool through an expert system able to increase the support level for the operators during maintenance operations. The process currently used by the tool first starts with questions about the general motion of the crane followed by more and more specific ones related to the subsystem motion limitation, the failed components and the possible causes. Failed components and causes are specified in 61 matrices information created based on Failure Mode and Effect Analysis (FMEA).

In order to verify the technical feasibility of our solution from an industrial point of view, the matrix 1_1, relating to the block of the rotation movement, was considered as an example (Figure 1). The degradation of this motion can be caused by 35 components and their corresponding 134 failure modes. There are 13 questions associated with the visible symptoms that can be used to identify the failure. This matrix is comparable to the test dataset matrix information, where the character-attributes relationship becomes the failure-symptoms relationship. The developed algorithm aims to identify the specific failure starting from a set of observed symptoms. Having the troubleshooting matrix information in the form of the test dataset (Table 1), the initialization, learning and conclusion of a troubleshooting session follow the same procedure presented in Section 3.4. For initialization, the set of probability replaced the yes/no answer in the matrix. To end the troubleshooting session, a probability threshold equal to 0.5 was set (Figure 2).

The performance of the system was evaluated both in the simplest case of identifying the 35 possible failure components and in the more complex case of identifying the 134 failure modes associated with them. Thus, 35 sessions were completed at the component level and 134 sessions at the failure mode level.

5. Results of the industrial case study

The simulation results (Figure 3) for the detection of faulty components only show that 71% of the components can be identified by correctly answering all the posed questions, while the remaining 29% of the possible failed components cannot be identified with the initial matrix information. The performance in the case of identifying failure modes

Fig. 3. Simulation results of industrial case study.

should be lower considering the unbalanced question/answer ratio. In fact, the following results are obtained:13% of the failure mode can be identified with the initial knowledge, while 44% is slightly uncertain and 43% is highly uncertain. In particular, the group of low uncertainty represents the failure modes that are not distinguishable from less than 8 other failure modes, and the group of high uncertainty represents the failure modes that are not distinguishable from more than 8 other failure modes.

The simulation results with only 13 questions gave less satisfying results. It was then decided to enhance the dataset with more questions to verify a better performance of the algorithm. An extensive effort for increasing the number of questions from 13 to 42 to reduce the uncertainty were carried out. The question extension was mainly related to specific symptoms of the possible damaged components. With the updated matrix information, the 13% identified failure group was improved to 41%, and the high uncertainty group was reduced from 43% to 17%, while no improvement was observed for the low uncertainty failure modes.

It is evident that a better performance is achieved by increasing the questions to problems ratio. The algorithm shows good applicability from the industrial viewpoint when this ratio is balanced and hence an adequate information matrix. To reach a good performance, a continuous update of the information matrix would be necessary to maintain a good question to problems ratio; however, the update can be an excessively burdensome operation. A future evolution that would make the solution more feasible from the industrial viewpoint is to develop the automatic generation or semi-automatic techniques of generating questions. Note that further improvements could be achieved by refining the set of probability associated with the information matrix, which was in this case defined a priori and limited to two values, namely 0.9 and 0.1, assigned based on the consistency with the answers provided by the user. Indeed, more detailed information concerning the components and the probability of each failure mode allows the algorithm performance improvement by assigning a more refined set of probability, e.g., higher probability for the most common failures, instead of using a uniform distribution. However, in this specific case, this kind of information was either unavailable or not accurate enough to be reliable.

6. Conclusions and future works

In the present work, a robust troubleshooting system was developed. An analysis of the required database, namely, the information matrix, was carried out as guidance to achieve the expected performance. Moreover, the system was tested in both test conditions and an industrial environment from a crane manufacturing company, where it is aimed to identify both the component and the failure mode that causes a crane downtime. Although the identified industrial case is related to a manufacturing company of hydraulic cranes, the algorithm can be easily adapted to different industrial contexts. Moreover, the system can be exploited both to support operators during failure diagnosis as well as to remotely assist customers. It should be noted that the tool performance is highly dependent on the quantity and quality of the questions available. Indeed, this work shows that the system knowledge extension allows to significantly

improve the performance of the troubleshooting process. However, adding questions may require an excessive effort that companies are hardly willing to dedicate. In this view, an automatic mechanism that can accommodate the user's suggestions and support technical experts expanding the system knowledge has been conceived. More specifically, Natural Language Processing techniques will be exploited to automatically process the incoming information since textual interaction is foreseen. Furthermore, a strategy to optimize the troubleshooting process will be studied to obtain the same performance with a fewer number of questions. These improvements are currently under development In the framework of the European funded AI REGIO project. [10]

References

[1] van Melle, W. (1978). Mycin: a knowledge-based consultation program for infectious disease diagnosis. *International Journal of Man-Machine Studies*, 10(3), 313 – 322.
[2] Bloch, H.P. and Geitner, F.K. (1999). *Chapter 12 - Knowledge-Based Systems for Machinery Failure Diagnosis*, volume 2 of *Practical Machinery Management for Process Plants*. Gulf Professional Publishing.
[3] Russo, D., Birolini, V., and Ceresoli, R. (2016). Fit: A triz based failure identification tool for product-service systems. *Procedia CIRP*, 47, 210 – 215. Product-Service Systems across Life Cycle.
[4] Pelc, A. (2002). Searching games with errors - fifty years of coping with liars. *Theoretical Computer Science*, 270, 71–109.
[5] Gill, J. and Wu, W. (2010). Twenty questions games always end with yes.
[6] Burgener, R. (2005). Artificial neural network guessing method and game.
[7] Suresh, S.R. (2017). A bayesian strategy to the 20 question game with applications to recommender systems.
[8] Jedynak, B., Frazier, P., and Sznitman, R. (2012). Twenty questions with noise: Bayes optimal policies for entropy loss. *Journal of Applied Probability*, 49, 114–136.
[9] MacKay, D.J.C. (2002). *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, USA.
[10] www.airegio-project.eu