Chatbots for Troubleshooting: A Survey

Camilo Thorne

IMS Stuttgart, Germany

camilo.thorne@ims.uni-stuttgart.de

Abstract. Chatbots are artificial agents designed to engage in text-based conversations (chats) with end users. They can be seen as a restricted kind of dialog system that deals with written rather than spoken language. They have been an object of research in academia and industry since the early days of artificial intelligence and computational linguistics. This paper intends to survey a specific subtype of chatbots, troubleshooting chatbots, which are chatbots that aim at automating troubleshooting contact centers and services. Troubleshooting chatbots have inspired considerable interest in industry due to their cost savings potential.

1. Introduction

Chatbots (chat bots, chatter bots, chatterbots) can be seen as a restricted kind of task-directed dialog systems in which interaction is textual, rather than spoken. Chatbots have been developed and studied since the 70s and the early days of natural language processing. Winograd's SHRDLU (Winograd, 1972) can be seen as an early kind of chatbot. Since the mid-90's, and due first to the emergence and later to the explosion of live chat forums within the world wide web, interest has grown even further.

An important application domain for chatbots is that of (corporate) technical assistance or *troubleshooting* (Acomb et al., 2007). Chat forums are used by corporations to help their employees or their clients solve operational problems regarding the services and/or equipment (e.g., server configuration issues or hardware problems) they use or

provide. Automating troubleshooting chat forums can give rise to substantial corporate savings by replacing or complementing human operators with artificial agents (Acomb et al., 2007). Troubleshooting chatbots are built out of 3 main components:

- a natural language understanding (NLU) and/or generation (NLG) component,
- a dialog planner or manager (DM), and
- a dialog state (DS).

The DS keeps track of the information relevant for the dialog, while the DM drives the conversation (updates the DS and decides what to answer to the end user), based on the DS and current user input. The system may rely in *domain knowledge* in the form of a lexical resource, a database or an ontology, encoding domain or world knowledge to perform their tasks. The main goal of the dialog is to *complete* its DS –i.e., identified all relevant information required to generate an appropriate troubleshooting recommendation.

Building a chatbot for troubleshooting can prove challenging, for several reasons. Troubleshooting dialogs tend to be short and ambiguous. Furthermore, it is a domain where few readily available and expert-annotated corpora exist, making data-driven approaches (common in natural language processing) problematic. This has motivated several, alternative approaches –and combinations thereof– , that rely on rule-based, machine learning (classification), pattern-matching or information retrieval techniques.

The aim of this survey is to describe the current state-of-the-art of chatbots for the troubleshooting domain. Their features will be discussed within the broader context of chatbots and text-based dialog systems. It is structured as follows. In Section 2 we will give an overview of troubleshooting dialogs. In Section 3 we give an overview of chatbot architecture, and survey known systems that partially or fully implement such architecture. In Section 4 we recall the main chatbot evaluation techniques. Finally in Section 5 we draw some conclusions, and highlight the main challenges that lie ahead in this particular application domain.

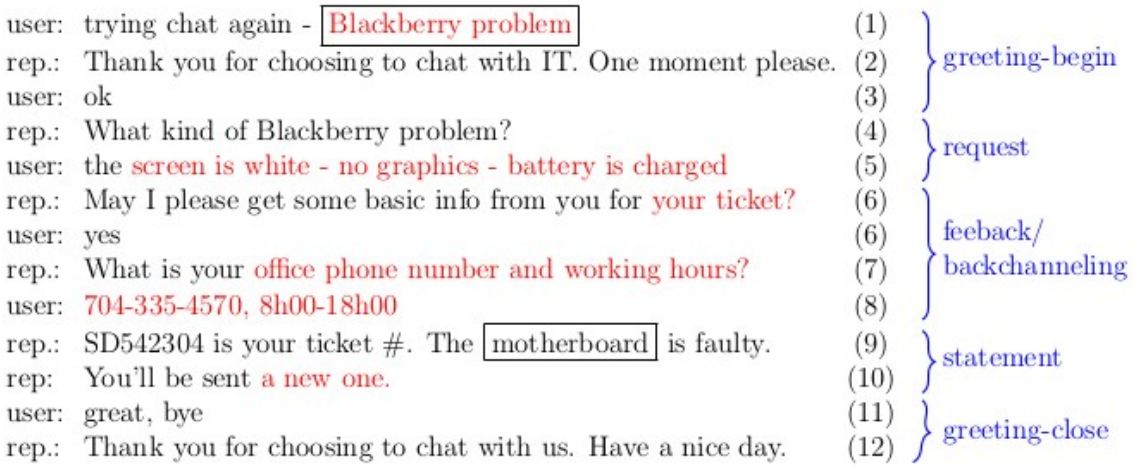

Figure 1. A typical troubleshooting dialog annotated with turn number (1–12) and dialog participant (user and IT operator or representative). In blue, the top-level dialog acts that structure the dialog. In red, the dialog entities. We enclose in a box the topic (the initial topic in turn (1) and, later, the refined topic in turn (9)) of the dialog (a Blackberry motherboard problem).

2. Troubleshooting Dialogs

Troubleshooting dialogs usually involve only two participants: the end user and a troubleshooting technician. Similar to generic dialogs, troubleshooting dialogs are structured as *turns*. That is to say, they consist of a sequence of alternating utterances made by each participant in the dialog. See Figure 1. Such dialog is usually of a *mixed-initiative* nature, that is to say that end-user and technician participate equally in the dialog[1]. They are *task directed*, as they have a concrete goal or purpose to fulfill: to determine a solution for a technical problem.

The *topic* or subject matter of troubleshooting dialogs is thus a technical *problem*, to which the user oftentimes refers only implicitly (Boye, 2007). Once a topic or problem has been introduced, additional information tends to be discussed. This additional information (known as the *focus*) often refers to the problem's *symptoms*. Topics may *shift* as the dialog progresses, although this is rare compared to generic dialogs. Troubleshooting dialogs instead show frequent changes in focus. Focusing allows the refinement of the dialog's problem until the concrete problem of the user is identified, and an answer (a solution) is returned, in a process known as *grounding*.

1) By contrast, in *single-initiative* dialogs, most utterances are made by one of the participants who is said to "drive the dialog".

Grounding is the key part of both generic and troubleshooting dialogs (Roque and Traum, 2009), and involves *reasoning*. As topics and foci may be referred to only implicitly by dialog participants, reasoning (either deductive, inductive or abductive) on background knowledge is required to make them explicit (Roque and Traum, 2009). For example, a common (IT) troubleshooting problem is "modem wrongly configures". End users typically refer to this problem via alternate descriptions (paraphrases) such as "can't connect to the Web" (Acomb et al., 2007). In such cases, reasoning on background knowledge about IT networks is required to properly resolve or disambiguate such paraphrases.

Each turn in a dialog conveys a communicative goal (intention) known as a *dialog act* (DA), which structures the dialog further. This highlights a specific feature of troubleshooting dialogs: their relatively uniform structure w.r.t. DAs. Indeed, turns can be grouped in terms of a relatively limited and stable set of DAs (Boye, 2007; Acomb et al., 2007), which tend to follow a clear order centered around grounding.

As the reader can see in Figures 1 and 2 (left), the overall (coarse-grained) structure of a troubleshooting dialog tends to be quite simple. An greeting-begin DA (e.g., a greeting) will be followed by a question or request of feedback on how to solve a problem (DA request). The key DA is the feedback-backchanneling DA, that refers to the grounding of the dialog. Once this is done an answer will be returned (the inform DA). Finally (the greeting-close DA) the technician will say user goodbye to the user forward him to some other service or opertator if grounding fails.

	DA	Frequency
greeting-begin	greeting-begin	$< 1\%$
request	request	$\sim 3\%$
feedback/backchanneling	feedback/backchanneling	$\sim 26\%$
inform	inform	$\sim 49\%$
greeting-close	greeting-close	$\sim 1\%$

Figure 2. Prevalent DA structure of troubleshooting dialogs (Acomb et al., 2007), in the VERBMOBIL DA notation (Jan Alexandersson et al., 1997). Right: Distribution of the respective DAs in the open- domain (telephone conversation) Switchboard corpus (Stolcke et al., 2000), also in VERBMOBIL format.

3. Troubleshooting Chatbots

3.1 Chatbot Architecture

In this subsection we describe the main architectural features of task-directed chatbots and troubleshooting chatbots. Chatbots are a restricted kind of dialog system, that mimics the behavior of a human participant in a natural language *textual* dialog or *chats*.

More concretely, they aim at mimicking and replacing human operators in task-directed chats, in order to lower the operating costs of the underlying service.

The general, high-level architectural design of chatbots is depicted in Figure 3. Depending on the domain however, several variations of this architecture are possible, due to the different structure of dialogs, thus constraining the behavior of the chatbot's single components. The main goal of a chatbot is hence to *complete* its internal information state, thus collecting sufficient information for fulfilling their intended task.

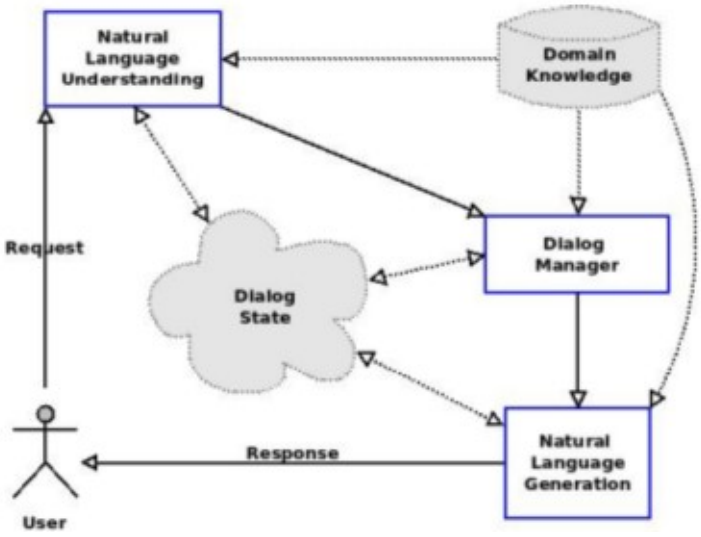

Figure 3. Chatbot high-level architecture.

Dialog State (DS)

The *dialog state* (DS) stores the *dynamic* information of the dialog. It stores the dialog's topic, the discourse referents (for anaphora resolution), information regarding the participants, and data from the system session. In other words, all the pieces of information necessary to properly understand user input and generate adequate system turns, but may change in the course of the dialog. The DS is typically stored in some frame data structure (e.g., a hash table, a record structure, a dictionary, etc.), and is designed to be continuously updated throughout the duration of the conversation.

In the case of troubleshooting chatbots, the DS must store three key pieces of information, namely:

- the current problem (dialog topic),
- the current problem's symptoms (dialog foci) and

- the current dialog act,

Additionally, extra information deemed useful for solution identification may be stored (e.g., the user's name, his location and contact details, etc.).

Dialog Manager (DM)

The *dialog manager* (DM) is the key component of every chatbot –and more in general every dialog system. It is usually defined as an iterative decision-making procedure, that decides on the basis of NLU input and the current DS what is the appropriate system's response, preserving at the same time DS consistency.

Dialog management strategies can vary widely between domains. Recalling what we saw in the previous section regarding troubleshooting dialog structure, the DM of troubleshooting chatbots must perform 3 basic tasks in order to efficiently drive (or mimic) troubleshooting dialogs:

1. **Initial problem identification:** during the initial phases of the dialog, a candidate problem frame is identified and the dialog state initialized; this implies:
   - initializing the dialog act(s) of the user turn(s),
   - initializing the name of the user, and
   - initializing the topic and focus of the turn(s).
2. **Grounding:** a conversation is engaged to gather more information, during which the DS is iteratively updated until completion or dialog failure. At each step, dialog *cohesion* must be preserved, viz., anaphora and ellipsis resolved, while topic and foci are updated. To deal with cohesion, the DM needs to *reason* over domain knowledge, contextual knowledge, in the form of external knowledge sources, such as ontologies (e.g., problem taxonomies) or databases (e.g., LDAP intranet databases). An important form of reasoning that the system has to support (see Boye, 2007) is *abduction* –reasoning from effects to causes–, as the system needs to identify or refine a problem (dialog topic) from its symptoms (dialog foci).
3. **Solution generation:** If the grounding succeeds, the system generates a solution. Otherwise, it forwards the user to a human operator.

NLP Resources

Last, but not least, to support NLU and NLG, text-based dialog systems and chatbots alike tend to leverage NLP techniques and resources to understand user input and generate a response. As these tasks are quite generic in nature, no further troubleshooting-specific architectural requirements are needed when designing a troubleshooting chatbot. We list the main ones:

- Morphological and syntactic annotation to prepossess text input (lemmatization, part-of-speech tagging, dependency parsing).
- Semantic processing (word sense disambiguation, named entity recognition, anaphora resolution, relation extraction, etc) to further analyze user input, and build and maintain the dialog state.
- Sentence planning, to dynamically generate a textual, grammatically correct response from the current DS.

Discussion

In practice, chatbots for troubleshooting and other domains implement only partially the architecture described in this section, depending on their specific use-cases. For instance, if the chatbot needs only to answer a question or engage in very short dialogs, keeping a dialog state or reasoning over contextual knowledge need not be maintained and a simpler and computationally cheaper DM can be used. This has led to a number of DM standards.

3.2 System Overview

System	Dialog state	Manager type	Domain knowledge	Initiative
Boye (2007)	Yes	Rule-based	Knowledge base	?
Chakrabarti and Luger (2012)	Yes	Rule-base	Knowledge base	?
Sankar et al. (2008)	no	AIML-based	Knowledge base	System
Acomb et al. (2007)	no	Classifer (maximum entropy)	Knowledge base	Mixed
Watson for IT (cf. Ferruci et al, 2010)	no	IR-based	Knowledge base and database	System

Table 1. Above, rule-based troubleshooting chatbots. Below, non-rule-based systems. By "Domain knowledge" we mean whether contextual information (i.e., reasoning over structured domain knowledge sources) is used by the system. The question mark indicates

that the system description contains no information regarding the feature, making it unclear if it is supported.

Each DM standard has a number of advantages and disadvantages that we outline next. These standards do not necessarily require keeping a DS or dealing with grounding. Table 1 gives an overview of the architectural components of the main troubleshooting chatbots we examined for the purposes of this survey.

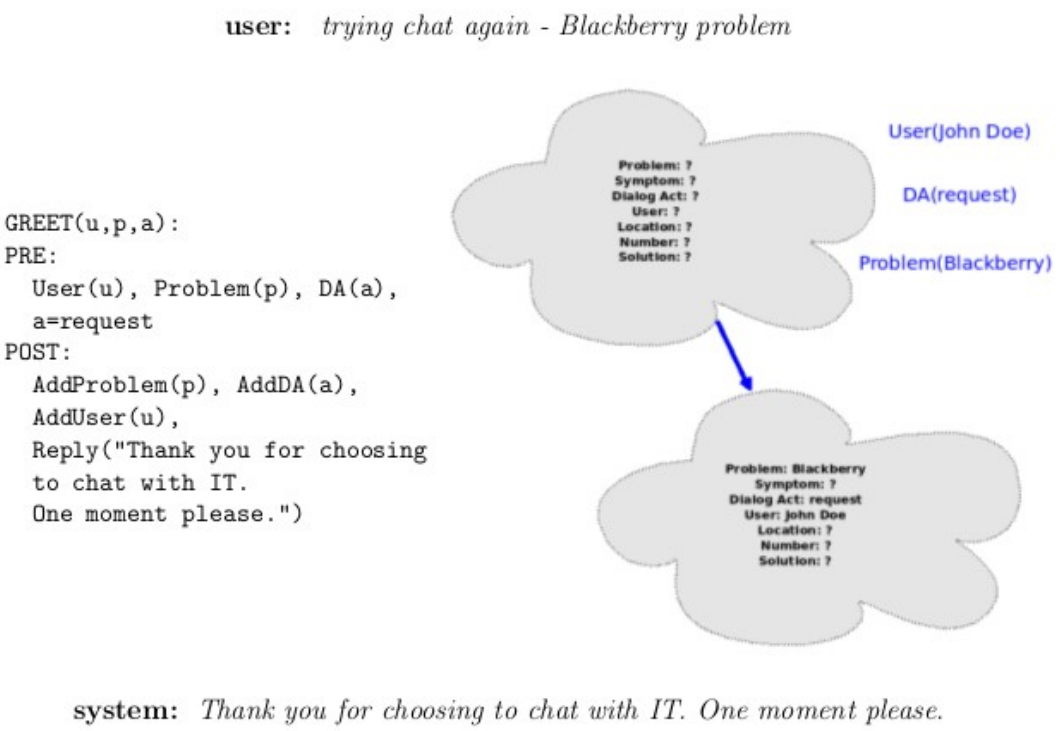

Figure 4. Applying the rule displayed on the left hand side to turn 1 of the conversation in Figure 1, will induce a modification (update) of the dialog state (empty until now), and trigger a sequence of actions, among which to utter turn 2 (i.e., to "reply").

Rule-based Systems

Chatbots driven by rules are common in the troubleshooting domain (Boye, 2007; Chakrabarti and Luger, 2012) and beyond (Banks et al., 2013; Janarthanam et al., 2013; Forbell et al., 2013; Sansonnet et al., 2012). Rule-based dialog management is depicted by Figure 4. Preconditions (PRE) check for the current state *S* of the dialog (in the example: the problem, the symptom(s), the DAs, the user, his location, his phone number). Actions (POST) induce a change in *S,* giving rise to a new state *S′*. Variables correspond to the fields of the DS. Rules can be seen as a "function" that *updates* DS *S* to *S′*. Moreover, updates will keep triggering successive rules and updates (involving potentially several rules at a time) as

long as their preconditions are satisfied. This will give rise to a sequence of actions and updated DS states known as a *plan* (Boye, 2007; Gibbon et al., 2000).

An important advantage of rule-based systems over other kinds of systems is that rules are not bound to speech input. Rule preconditions can react to DS fields derived from sources other than user turns, and actions can generate many kinds of outputs beyond speech. This is achieved via "side effects", i.e., remote calls to domain knowledge. This allows seamless integration of external knowledge and of reasoning services (cf. Gibbon, Mertins and Moore, 2000), which are key to grounding troubleshooting dialogs (specifically: for abduction).

The rules, however, need to be specified manually, and usually will not generalize to other domains (cf. Gibbon et al., 2000), let alone troubleshooting subdomains. Also, rules presuppose a relatively low variability in dialog structure and may fail to fire if the end user deviates from the DMs' expected input.

Artificial Intelligence Markup language (AIML)

AIML is a markup language used to declare pattern-template pairs know as *categories*. A *pattern* is a regular expression used to match user input. Its associated *template,* represents a system response. Dialog management is straightforward: whenever the a pattern is positively matched to user input, the corresponding template is returned.

AIML is based on XML. See Figure 4. A single AIML file can be used to specify all system-user interactions or turns, and can be crafted for any application domain. AIML has become a *de facto* standard for developing simple chatbots, and has given rise to a number of chatbot open source APIs, among which the ALICE (Artificial Linguistic Internet Computer Entity) architecture[2] and its sibligs (Sankar et al., 2008).

The main advantage of AIML is that it is easy to use, configure and deploy, and is well supported by chatbot development APIs[3]. It does not require large computational resources. AIML chatbots lack however a dialog state (they are *reactive* agents), and are unable to support grounding and hence to fully understand troubleshooting dialogs. Also, an AIML script category has to be manually written for *every* possible user-system turn pair, as no linguistic processing other than regular expression matching is supported natively by the standard.

2) http://www.alicebot.org/aiml.html

3) https://developer.pandorabots.com

```
<aiml>
 <category>
   <pattern>*BLACKBERRY*</pattern>
   <template>
     what kind of blackberry problem?
    <set name="topic">blackberry</set>
    <think><set_ip>134.155.214.56</set_ip></think>
   </template>
 </category>
 <category>
   <pattern>*</pattern>
   <that>WHAT</that>
   <template>anything else?</template>
 </category>
</aiml>
```

Figure 4. Sample AIML script. The <think>...</think> tag is used to declare side- effects (in this example: to set or retrieve the user's address for the system's back-end).

Information retrieval (IR)

In some specific use-cases, in particular when end users are interested only in issuing questions, chatbots can leverage on information retrieval (IR) techniques. A known example for the troubleshooting domain is Watson for IT (Ferrucci et al., 2010), a question-answering system for troubleshooting IBM products[4], which searches for answers over very large technical documentation corpora, and relies (partly) on IR libraries (such as, e.g., Lucene[5]). The idea is first to search for and extract candidate answers to end-user troubleshooting questions, and then use linguistic processing (morphological and syntactic analysis, semantic annotation, databases, knowledge base reasoning) to *rerank* candidate answers. The lack of a dialog state however makes it difficult to adequately process full-fledged troubleshooting dialogs.

Classification

Rule-based techniques tend to overfit the problem domain, cannot be readily ported to other domains, and their performance does not improve over time. In domains where large corpora of annotated troubleshooting dialogs such performance gaps can be closed via

4) It was obtained by adapting the original Watson system, built to play the Jeopardy quiz show, to IBM product troubleshooting.

5) https://lucene.apache.org/

supervised approaches (Acomb et al., 2007). Their main features are two: **(a)** their DM component is built around supervised classification, and **(b)** they rely on annotated gold corpora for training.

The idea here is to *predict,* given some user input, all or some of the fields of the dialog state by learning a statistical classification model (mainly logistic or maximum entropy models) from domain-specific dialog corpora where DAs, topics, foci, etc., have been annotated by domain experts or linguists. This can yield reasonable performance depending on the task and the quality of the gold standard. For example, Acomb et al. (2007) trained a maximum entropy model to recognize the DA of user turns, achieving close to 80% accuracy. This approach, which works well for open domain dialogs (Stolcke et al., 2000) is limited in the troubleshooting domain by the availability and quality of annotated dialogs, which tend to be scarce and of low quality when available[6].

Approach	**Advantages**	**Disadvantages**
Rule-based	Keeps a dialog state, can build up plans and integrate reasoning services	Rules need to be hand crafted, systems cannot be ported to other troubleshooting domains
AIML	Simple design, based on a mature standard (XML)	No dialog state, scripts need to be hand crafted
Classifier	Can keep a dialog state and adapt to troubleshooting domains (if data is available)	Dearth of annotated troubleshooting dialog to train classifiers
IR-based	Good performance answering troubleshooting questions	Not designed to deal with multiturn dialogs or handle grounding

Table 2. Advantages and drawbacks of the different approaches to dialog management followed by the troubleshooting chatbots mentioned in this survey.

6) Most datasets are proprietary, when available.

Discussion

The current state-of-the-art in troubleshooting chatbots exhibits a number of positive characteristics. However, these advantages are counterweighted by a number of drawbacks (see Table 2). As a result few, if any, are capable of fully adhering to the design principles and features outlined in Section 3.1., decreasing their intended robustness.

Indeed, state-of-the-art dialog systems and chatbots (Williams et al., 2013; Gibbon et al., 2000) rely on *mixed* approaches to dialog management that combine rules, probabilistic reasoning and classification, to *maximize* system robustness. The main techniques are two: **(a)** to use classification to partially fill DS fields and then apply rule-based planning as described previously, and **(b)** to use probabilistic planning techniques in the form of so-called (partially observable) Markov decision procedures which can dynamically adapt DM rules to changes in dialog structure. Combinations of **(a)** and **(b)** induce further gains in robustness.

Examples of chatbots resorting to data driven or mixed approaches abound in the literature. Negi et al. (2009) combine AIML and classification, to design chatbots that not only match patterns but can also *predict* them, by using the conversation logs to bootstrap and reinforce pattern prediction from user turns. This makes it easier to design multiple and cross-domain AIML chatbots as now only a subset of all potential system-user turns needs to be explicitly covered by AIML categories. Another interesting chatbot is BOB (Kirschner and Bernardi, 2010), where a planning- or rule-based chatbot dealing with library reservation and inquiries, was coupled with a logistic classifier to predict not only DS fields, but also rules. To collect (seed) gold data for system bootstrapping, a so-called Wizard-of-Oz experiment[7] was conducted.

4. Evaluation Strategies

Evaluation Metrics

Historically, the oldest known evaluation technique for chatbots and dialog systems is the so-called *Turing test* (Mauldin, 1994). The Turing test is an evaluation methodology derived from the fields of artificial intelligence (AI) and cognitive systems. It was first proposed by Turing in the 1950s (cf. Turing 1950), and has been applied to chatbots since the 1980's, in the context of the so-called Loebner prize competition (Loebner, 2009). The overall idea of the test is to make a human tester interact (unknowingly) with a dialog system, to measure **(a)** how many conversations turns he/she takes to identify it as a dialog system, and **(b)** how many system turns match the answers of an (arbitrary) human agent.

7) In such an experiment, a live chat with a human operator that poses as a chatbot is conducted over a number of days, to collect gold standard chat data.

Over time, however, more robust evaluation strategies have arisen (Shawar and Atwell, 2007; Glass, 1999; Walker et al., 1997). These evaluation strategies measure different aspects of a chatbot's functionality (regardless of their domain) and behavior..

1. **User Satisfaction.** Since chatbots aim at providing and automating conversational front-ends to a number of services such as (IT) troubleshooting, it is also possible to evaluate them using method derived from the human-computer interaction (HCI) literature (Dzikovska, Steinhauser, and Campbell, 2011).

   - The main measure considered is the *user satisfaction* (US) metric that estimates (e.g., in a scale of 1 to 5) the impression a system made on end users (Sankar et al., 2008). It is usually collected as follows: **(a)** obtain numbers via user surveys, and **(b)** consider a (statistically) representative sample of users.

   - US results can be further refined by considering *agreement* (Ultes, Schmitt, and Minker, 2013). US scores are discrete scores. Assigning a score to a chatbot can be thus seen as a kind of (discrete) "annotation" task, making it possible to apply Cohen's/Fleischer's κ *kappa coefficient* (that measures annotation agreement) to measure the agreement on scores defined by

$$\kappa = \frac{\overline{P}_a - \overline{P}_e}{1 - \overline{P}_e} \tag{1}$$

     that measures how much the observed agreement differs from random agreement. The higher the kappa, the greater the agreement.

   - Finally, one can run an statistical test (such as a Student t-test) to understand whether the system shows an statistically significant gain in US w.r.t. to some US baseline (Ultes, Schmitt, and Minker, 2013).

2. **Precision, Recall and Accuracy.** Another family of evaluation metrics that have been applied to (IR-driven) chatbots (and dialog systems) are the IR measures of precision, recall and accuracy (Molino et al. 2013; Ferrucci et al. 2010; Acomb et al. 2007). Such measures quantify how well the chatbot is able to select appropriate system follow-ups (answers) to user turns from a space of potential candidates (answers).

   - Dialog *precision* measures the rate of correct vs. incorrect turns within a conversation

$$P = \frac{correct\ answer\ turns}{true\ answer\ turns + false\ answer\ turns} \quad (2)$$

- Dialog *recall* measures the rate of correct vs. expected correct turns (as per the systems' knowledge) within a conversation

$$R = \frac{correct\ answer\ turns}{true\ answer\ turns + false\ turns} \quad (3)$$

- Dialog *accuracy* measures the rate of correct turns within a conversation

$$A = \frac{correct\ answer\ turns}{total\ turns} \quad (4)$$

3. **Task Completion.** The metric of task completion measures the percent of DS completion observed at the end of a user-system dialog (Negi et al. 2009; Schooten and Akker, 2011). This metric is particularly appropriate for task-directed dialogs or chats (e.g., booking a flight, buying an item, filling an online form). Intuitively, it quantifies the capacity of the chatbot to collect sufficient information for the dialog (i.e., the grounding) to succeed and generate a solution.

System Overview

Few, if any, systems mentioned in Table 1 report any evaluation along the lines described in this section. Table 3 summarizes the best performances of those systems that do report an evaluation: the system by Acomb et al. (2007), and Watson for IT. No system reports kappa figures, statistical significance tests, or Turing test evaluations. This makes its difficult to assess or compare their performance, not least, with respect to chatbots for other domains that implement mixed strategies[8].

8) E.g., Negi et al., (2009), a multiple-domain chatbot system that combines AIML with classification, reports an average of 74,2% task completion across 5 different tasks (in domains such as the clinical domain). But is unclear if any of the troubleshooting chatbots reviewed in this paper can come anywhere close to this number.

This lack of evaluation results is due to two factors: **(1)** Scarcity of gold standard troubleshooting dialog datasets, that are key for IR-based metrics, and for system cross-evaluation. **(2)** Troubleshooting chatbots tend to be the result of research-industry cooperations, meaning that many cooperation outcomes, among which evaluation results, do not reach the public domain.

System	**User satisfaction**	**IR measures**	**Task completion**
Watson for IT (Ferrucci et al., 2010)		70% (global *P*)	
Acomb et al. 2007		62.3% (global *A*)	

Table 3. Performance statistics for those systems that report an evaluation (the other systems reported no evaluation). Blanks mean that the metric is not applicable to the system. For Watson we report performance for the 2010 Jeopardy dataset.

5. Conclusions

In this paper we have provided an overview of the state-of-the-art on chatbots for the troubleshooting domain. Chatbots are text-based dialog systems that aim at automating chats. Troubleshooting chats and dialogs deal with the resolution of technical problems. Troubleshooting chatbots aim at partially automating troubleshooting service centers via text-based dialogs. We have outlined the key properties of troubleshooting dialogs, of troubleshooting chatbot architectures, reviewed a number of recent troubleshooting chatbots described in dialog system literature, and discussed the main evaluation strategies currently available for such chatbots.

Troubleshooting dialogs are task-directed and exhibit a rather standard turn structure, where grounding and reasoning (e.g., abduction to problems from symptoms) play a key role. Troubleshooting chatbots exploit this standardized structure in various ways.

Some chatbots incorporate a dialog state that stores key pieces of information for troubleshooting dialog semantics: dialog topics (problems), foci (symptoms) and context (domain knowledge and external information sources). This dialog state is thereafter used to drive pattern- or planning-based dialog managers. Other chatbots rely instead on data-driven strategies, namely turn or answer search, or, alternatively, turn prediction. They all

share the following positive feature: they are all built on top of known and rather mature standards and chatbot APIs.

Troubleshooting chatbots face still a number of significant challenges. Pattern- and planning-driven systems suffer from low domain adaptability, as rules and patterns need to be hand-crafted. Whereas data-driven approaches suffer from a scarcity of troubleshooting dialog gold standards for training. This makes it moreover difficult to experiment with theoretically more robust *combined* approaches, drawn from dialog system and chatbot literature.

Last, but not least, this scarcity of gold standards makes it very hard, if not almost impossible at the current state, to understand and compare the performance of the different approaches and systems, and may also explain why so few troubleshooting chatterbots report evaluation results.

Acknowledgments

The author thanks his colleagues and the anonymous reviewers for their comments to earlier drafts of this paper. Part of the work in this survey was carried out while the author was working as dialog systems researcher at the Center of Advanced Studies of IBM Italia (Italy).

References

Acomb, K, Bloom, J., Dayanidhi, K., Hunter, P., Krogh, P., Levin E., and Pieraccini, R. (2007). Technical Support Dialog Systems: Issues, Problems and Solutions. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*.

Alexandersson, J., Buschbeck-Wolf, B., Fujinami Tl., Maier, E., Reithinger, N., Schmitz, B., and Siegel, M. (1997). *Dialogue acts in VERBMOBIL-2*. Technical report, DFKI Saarbrucken.

Banks, R. E., Jiang, R., Kim, S, Niswar A., and Hui Yeo, K. (2013) AIDA: Artificial Intelligent Dialogue Agent. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2013)*.

Boye, J. (2007). Dialogue Management for Automatic Troubleshooting and Other Problem-Solving Applications. In *Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue*.

Chakrabarti, Ch., and Luger, G. F. (2012). A Semantic Architecture for Artificial Conversations. In *Soft Computing and Intelligent Systems and 13th International Symposium on Advanced Intelligent Systems (SCIS-ISIS 2012)*.

Dzikovska, M. O., Moore J. D., Steinhauser, N., and Campbell G. (2011). Exploring User Satisfaction in a Tutorial Dialogue System. In *Proceedings of the 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2011)*.

Ferrucci, D., Brown E., Fan J., Chu-Carroll, J., Gondek, D., Kalyanpur, A. A., Lally A., Murdock, J. W. (2010). Building Watson: an Overview of the DeepQA Project. *AI Magazine* 36 (3): 59–79.

Forbell, E., Kalisch, N., Morbini, F., Christoffersen, K., Sagae, K., Traum, D., and Rizzo, A. A. (2013). Roundtable: an Online Framework for Building Web-Based Conversational Agents. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2013)*.

Gibbon, D., Mertins, I., and Moore R. K. (2000). *Handbook of Multimodal and Spoken Dialogue Systems: Resources, Terminology, and Product Evaluation.* Springer.

Glass, J. R. (1999). Challenges for Spoken Dialogue Systems. In *Proceedings of the 1999 IEEE ASRU Workshop*.

Janarthanam, S., Lemon, O., Liu, X., Bartie, P., MacKaness, W., and Dalmas, T. (2013). A Multithreaded Conversational Interface for Pedestrian Navigation and Question Answering. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2013)*.

Kirschner, M., and Bernardi, R. (2010). Towards an Empirically Motivated Typology of Follow-up Questions: the Role of Dialogue Context. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2010)*.

Loebner, H. (2009). How to hold a Turing Test contest. In *Parsing the Turing Test* (pp. 173-179). Springer.

Mauldin, Michael L. 1994. "CHATTERBOTs, TINYMUDs, and the Turing Test. Entering the Loebner Prize Competition." In *Proceedings of the 12th AAAI Conference on Artificial Intelligence (AAAI 1994)*.

Molino, P., Basile, P., Santoro, C., Lops, P., de Gemmis, M., and Semeraro, G. (2013). A Virtual Player for "Who Wants to Be a Millionaire?". In *Proceedings of 13th International Conference of the Italian Association for Artificial Intelligence* (*AI*IA 2013)*.

Negi, S., Joshi, S., Chamalla, A., and Subramaniam, L. V. (2009). Automatically Extracting Dialog Models from Conversation Transcripts. In *Proceedings of 9th IEEE International Conference on Data Mining* (*ICDM 2009)*.

Roque, A. and Traum, D. (2009). Improving a virtual human using a model of degrees of grounding. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence (IJCAI 2009*).

Sankar, G. R., Greyling, J., Vogts, D., and du Plessis, M. C. (2008). Models Towards a Hybrid Conversational Agent for Contact Centres. In *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*.

Sansonnet, Jean Paul, Daniel Werner Correa, Patricia Jacques, Annelies Braufort, and Cyril Verrechia. 2012. "Developing Web Fully-Integrated Conversational Assistant Agents." In *Proceedings of the 2012 ACM Research in Applied Computation Symposion (RACS 2012).*

van Schooten, B., and op den Akker, R. (2011). Vidiam: Corpus-Based Development of a Dialogue Manager for Multimodal Question Answering. In van den Bosch, A., and

Shawar, B. A., and Atwell, E.. (2007). Different Measurement Metrics to Evaluate a Chatbot System. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*.

Stolcke, A., Coccaro, N., Bates, R., Taylor, P., Ess-Dykema, C. V., Ries, K., Shriberg, E., Jurafsky, D., and Martin, R. (2000). "Dialogue act modeling for automatic tagging and recognition of conversational agents". Computational Linguistics, 26(6):339–373.

Turing, A. 1950. Computing Machinery and Intelligence. *Mind* 59: 433–460.

Ultes, S., Schmitt A., and Minker, W. (2013). On Quality Ratings for Spoken Dialogue Systems – Experts Vs. Users. In *Proceedings of the 13th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*.

Walker, M. A., Litman, D. J., Kamm C. A., and Abella A. (1997). PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1997)*.

Wang, Y. F., and Petrina, S. (2013). Using Learning Analytics to Understand the Design of an Intelligent Language Tutor – Chatbot Lucy. *International Journal of Advanced Computer Science and Applications* 4 (11): 214–131.

Williams, J. D., Ramachamdran, D., and Black, A. C. (2013). The Dialog State Tracking Challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2013).*

Winograd, T. (1972). *Understanding Natural Language*. Academic Press.