Bottester : Testing Conversational Systems with Simulated Users

Marisa Vasconcelos
Heloisa Candello
Claudio Pinhanez
IBM Research
Sao Paulo, Brazil
marisaav@br.ibm.com
heloisacandello@br.ibm.com
csantosp@br.ibm.com

Abstract
Recently, conversation agents have attracted the attention of many companies such as *IBM*, *Facebook*, *Google*, and *Amazon* which have focused on developing tools or APIs for developers to create their own chatbots. In this paper, we focus on new approaches to evaluate such systems presenting some guidelines resulted from evaluating a real chatbot use case. Testing conversational agents or chatbots is not a trivial task due to the multitude aspects/tasks (e.g., natural language understanding, dialog management and, response generation) which must be considered separately and as a mixture. Also, the creation of a general testing tool is a challenge since evaluation is very sensitive to the application context. Finally, exhaustive testing can be a tedious task for the project team what creates a need for a tool to perform it automatically. This paper opens a discussion about how conversational systems testing tools are essential to ensure well-functioning of such systems as well as to help interface designers guiding them to develop consistent conversational interfaces.

Author Keywords
Test; user interface; conversational interfaces; chatbots

Introduction
Conversational systems have been gaining popularity thanks to advances in artificial intelligence and in other technologies such as speech recognition. Chatbots and other text-based agents are being used in several domains such as customer service, education, finance advising, and others. Companies such as *Google*, *IBM* and *Facebook* are jumping into the development of platforms for building conversational interfaces and also integrating various apps into chatbots.

As any software system, testing is required to detect problems in the system and compare the current version with previous versions [5]. However, conversational interfaces are very complex and composed of many modules (e.g., dialog management, natural language processor) what makes testing not trivial because of the diversity of interaction parameters and also their interrelation and temporal dynamics.

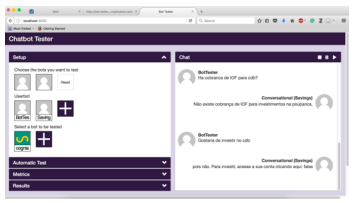

Figure 1: Bottester Interface

Furthermore, a chatbot evaluation is very dependent on the context of the application, for instance, an education chatbot has completely different test cases from a travel agent one. The goal of the chatbot is also an important aspect since task-oriented chatbots have specific goals which guide their interaction, while non-oriented chatbots do not have a goal and aim to establish free conversations with the users. Moreover, testing chatbots with human beings is a costly, time-consuming task, and tedious.

Previous work [2,3,5] have proposed metrics which are mostly interested in testing separately modules of a chatbot or have performed qualitative evaluations with small groups of people [1,3] to assess user satisfaction. The famous *Loebner* competition has also been used to evaluate the ability of chatbots to have human-like conversations.

We here explore the testing of chatbot systems focusing on the user perspective, that is, by observing the resultant interaction of all chatbot modules. Thus, we propose to simulate users with a chatbot tester tool which interacts with chatbots and collects measures about the interactions. This is the approach we explored in a tool we call *Bottester*.

This is an initial report about indicatives of quality and user satisfaction using metrics collected during interactions of the *Bottester* with the chatbot system. The testing system simulates a large number of interactions with the chatbot system, automatically creating dialogues which resemble real user interaction. The testing system then computes automatically metrics which are related to user satisfaction as well as some related to the overall performance of the system. As a use case, we experiment with a finance chatbot system **(*CognIA*)** implemented to help a user to make basic investment decisions.

CognIA Chatbot System

We developed a chatbot specialized in finance advice, named *CognIA*, which gives investment advice to people with limited finance knowledge. The design of *CognIA* was thought to be a mix of a free-conversation system and a goal-driven system which means that it does not have the sole objective to complete a task: it has to keep the user engaged and willing to come back for future interactions.

The *CognIA* implementation was designed as a multi-agent architecture composed of three chatbots: *Cognia* with acts as a moderator, *PoupancaGuru* which is responsible to advise about savings accounts and *CDBGuru* which is specialized in certificates of deposit. *CognIA* has a database consisting over 491 question-answer pairs in Portuguese language with question variations (e.g., *"Tell me about savings?" or "What is savings"*) summing up to 38 different intents.

The Bottester Tool

During the prototyping stage of *CognIA*, testing scenarios were created using the dialogues collected from a *Wizard of Oz* experiment. After that, the *CognIA* had its corpus of data increased manually to improve its natural language processing. In such stage, the chatbot system needed to be tested automatically because the space of possible test cases is too big to be explored manually. Thus, we needed a tool to perform and simulate users interacting exhaustively with the system.

The development of the *Bottester* tool for the *CognIA* system was therefore performed after most of the system was already implemented. The advantage of this approach was that to simulate the users we could use a corpus of data containing frequent questions and answers which was already collected and built.

Figure 2: Bottester Results Interface

Metric	Assessment
Mean answer size	Answers conciseness
Answers frequency	Knowledge base limitation
Word frequency	Vocabulary limitation
Number of (in)correct answers	Ability to understand the language
Mean response time	Overall user perception of the service quality
Response time per question	Delay of each question
Response time per agent	User perception of each agent capacity

Table 1: Metrics used by the Bottester tool

Figure 1 shows the web-based *Bottester* interface. Basically, the inputs for *Bottester* (left part of the interface) are a file containing all the questions to be submitted to the tested system, the respective expected answers, and the configuration parameters for the tests and the connection protocols to the chatbot system. There are optional parameters such as the number of times the input scenario is to be executed and a timeout which defines the maximum interval for the answer to be received. On the right, the interface shows the submitted questions which were sent to the chatbot after pressing the *play* button.

Evaluation Metrics

As mentioned earlier, the idea of the *Bottester* tool is to simulate a real user interacting with the system. Indeed, we are collecting our metrics at the interface level which means that we cannot point out which chatbot module may have a performance problem but rather identify the effects of that problem for the user interaction. The *Bottester* can be used during all different stages of the development phase, for instance, to compare different versions of the same chatbot system.

Initially, we focused on improving the content presentation of the tool and thus we start measuring how the answers were presented in the chatbot interface. For that, we collect the size (in characters or words) of each answer given by the chatbot. The metric indicates how verbose are the answers pointing to the project team which ones should be shortened. Conciseness is often important since the chatbot can be accessed through several types of devices (e.g. mobile phones) and thus performance is in many cases dependent on what the screen can display.

Moreover, we can test the accuracy of the chatbot system's natural language intent (speech-act) classifier if that information is available. Since we built the application, we have access to each bot agent knowledge base and then the ground-truth of all questions. In the current version of the tool, we consider that the answer is correct only if there is a total match between the answer and the expected answer defined in the input file. Eventually, other similarity metrics (e.g. perplexity and distance measures) will be implemented to account for partially correct answers. The number of correct and incorrect answers gives an idea of how relevant and appropriate is the response for each question, what indirectly assess the performance of the natural language processor module.

Orthogonally, we also measure the number of repetitive answers. A high number of repetitive answers can suggest that the bot has a limited knowledge base (e.g., if there is a significant number of *"I don't know answers"*) or may have a problem in their NLP/classification module. Figure 3 shows a word cloud created from the answers from CognIA. We can observe that the chatbot has the word *savings* in many of its answers.

Finally, the *Bottester* tool also collects the response time experienced by the user which is the time interval between the question submission and the response arrival. The response time is a key metric because it is often related to the user's perception of the quality of service. This measure is an aggregate evaluation since it depends on several aspects such as the chatbot system server capabilities and its load, network delays, and response processing at the user side. We are especially interested in identifying whether the delay exists and therefore if the user experience can be being im-

Example	Scenario
Cognia: Hello, how can I help you? *Bottester*: I would like to **invesg** on CDB.	typo testing
CDBGuru: Which amount would you like to invest? *Bottester*: **1000 sandwichs** *Cognia*: Sorry, rephrase your question	wrong answer
Bottester: I would like to try the savings fund *PoupancaGuru*: Please contact your bank.	wrong answer

Table 2: Example of dialog between the Bottester and CogniIA

Figure 3: Word cloud generated using CognIA answers

paired by that. Future features of the testing tool will allow identifying where the delay is, which will help developers to understand why the delay was caused and in which part of the code. On the other hand, fast response times are problematic since the user may not keep up with the feedback [3].

Testing *CognIA*

Initially, we performed a sanity check of the *CognIA* system verifying whether all questions and answers implemented in it were being responded. We mainly focused on the number of correct/incorrect answers but we also used the response time to identify what types of questions were taking a significant amount of time. We also separated some of the metrics per each of the three chatbot agents (see Figure 2) and also per type of question. This separation allows us to assess and correlate how the user experiences each agent. Table 1 summarizes the metrics used by *Bottester* and Figure 2 shows the visualization of the same metrics.

Currently, we are refining the set of questions used in the sanity check by generating typos in the words of each question to be sure the intent classifier is able to withstand simple variations in the way the user formulates questions. Moreover, we randomly select words from the answers and replace by their synonyms. We intend with those two classes of experiments to mimic better user interactions. Table 2 shows an example of dialog in which we test different kinds of errors.

Discussions and Future Work

Testing a chatbot is not trivial since the space of possible inputs is extremely large: each dialog is interactive which means that each conversation may happen only once. Manual testing is time-consuming, often inaccurate, and, in scenarios of repetition, a very tedious task for developers. Automated testing or performing tests using a tool is not only faster but also allow the coverage of an extensive number of scenarios. The idea is not to replace experiments with real users but to make the system robust enough and with no critical problems before user experiments. Automatic testing can easily detect conversation flow stoppers such as the bot answering "I don't know", nonsense statements, or repeating utterances, which clearly will affect the user experience. Knowing those issues, experience and content designers can identify and fix those faster and earlier in the process.

Some of the main issues to develop a chat testing tool are is to measure user satisfaction, task success, and dialog cost. This is even more problematic in non-task orientation situations, where it is challenging to define and predict the contents and topics of user utterances and therefore specify conversation scenarios for testing user satisfaction. Our testing tool can assist developers and designers in evaluating single and multi-agent chatbots in those contexts.

Our interest in this workshop is to discuss which metrics are more suitable or relevant for chatbots that we should include in the next version of our testing tool. It is also important to understand how the *Bottester* could be enhanced to handle different application scenarios and contexts. We also would like to discuss the best ways to show or visualize the results of this automatic tool.

References

1. Bayan Shawar and Eric Atwell. 2007. Different Measurements Metrics to Evaluate a Chatbot System. In NAACL-HLT-Dialog.
2. Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In EMNLP.
3. Karolina Kuligowska. 2015. Commercial chatbot: performance evaluation, usability metrics, and quality standards of embodied conversational agents.
4. Marilyn Walker, Diane Litman, Candace Kamm, and Alicia Abella. 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In EACL.
5. Michael McTear, Zoraida Callejas, and David Griol. 2016. Evaluating the Conversational Interface. Springer International Publishing, 379–402.