Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 2267–2274

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Algorithm Inspection for Chatbot Performance Evaluation

Vijayaraghavan V., Jack Brian Cooper, Rian Leevinson J.

Infosys Limited, India
Vijayaraghavan_V01@infosys.com

Abstract

With the usage of Chatbots growing at an unprecedented rate, it is imperative that they are thoroughly tested, verified and validated. This is done to ensure that Chatbots do not fail during operation. Chatbot failures are undesirable and they often occur when the bot is provided with ambiguous or illegible input. The Chatbots have to be thoroughly tested to ensure that they do not fail under any circumstance and have mechanisms to deal with such scenarios. Although various methods are in use to test Chatbots, algorithm testing is a promising solution to the problem. This involves the use of techniques such as cross-validation, grammar and parsing, verification and validation and statistical parsing. This paper aims to explore the prominent types of chatbot testing methods with detailed emphasis on algorithm testing techniques.

Keywords: Chatbot testing, algorithm testing, NLP, ambiguity, verification and validation, machine learning

10.1016/j.procs.2020.04.245

1. Introduction

Chatbots are a class of intelligent, conversational software algorithms activated by natural language input. They can intelligently respond to inputs, understand commands and execute tasks [1]. Although Chatbots have existed for a long time, they are getting better due to the availability of data, better computing power and open-source coding platforms. These factors have led to widespread implementation of Chatbots across domains and industries. From customer care to social media forums, we encounter Chatbots regularly in various forms.

Chatbots are generally designed to serve specific purposes. For example, customer care Chatbots are specifically designed to cater to the needs of customers who seek help while conversational Chatbots are built to have interactive communications with users. These Chatbots are designed and built based on the domain in which they will operate. Therefore, they may fail if they are operated in conditions unfamiliar to them or when ambiguities arise during operation.

Intelligent systems such as Chatbots are designed to mimic the performance of human experts in the specific domain. Often, the performance of the chatbot is measured by its similarity to that of a human being. Hence it is imperative that Chatbots operate and interact in a desirable manner [10]. Organizations have to rigorously test and check their Chatbot before it is released into production. This is mainly done to prevent unexpected bot failures. Chatbots tend to fail when they are presented with situations and inputs that they are not familiar with. This may include ambiguous statements, grammatically incorrect phrases and statements outside the operating domain of the bot. Current chatbot testing approaches either monitor the output of the chatbot to evaluate the performance or study the inner functionality of the chatbot and the various algorithms involved. The various testing techniques and algorithm specific inspection techniques are explored in the subsequent sections.

2. Chatbot Output Testing Techniques

Chatbot output testing involves giving inputs to the chatbot and monitoring its response. The chatbot's output is analyzed to determine the quality of the response, relevance, completeness, accuracy and context. This may be performed by letting users interact with the chatbot in a controlled manner and using the feedback to determine its performance. However, assessing the performance of a chatbot based on its interaction with a user could be extremely subjective. The measure may differ from case to case in terms of the context, domain and the type of interaction. Therefore, it is crucial that benchmarks or specific standards are defined to evaluate Chatbots in a standardized manner.

Chattest [2] is a collaborative open source project that offers 120 questions across various paradigm such as Answering, Error Management, Intelligence, Navigation, Onboarding, Personality and Understanding. This serves as a comprehensive platform to broadly assess the performance of Chatbots through different scenarios. The performance of the bot across these key areas can be used to improve the design and functionality of the chatbot. Moreover, cross-checking the performance using these techniques help identify cases of imminent failure. Developers can thereby identify specifically which areas of the chatbot are performing well, and where any issues need addressing.

Methods to assess the output of the chatbot include performing semantic analysis on feedback provided (i.e. whether reviews are positive or negative, and why this is the case) or by asking users to complete a satisfaction survey and performing statistical analysis on this, or simply even rating satisfaction on a response-by-response level. While these methods are convenient and provide valuable insights into the chatbot's performance, they do suffer from a lot of drawbacks. The first is that it is difficult to obtain sufficient user satisfaction or feedback data. An ideal scenario is to make the chatbot learn from user feedback by building a reinforced loop feedback system. However, the feasibility and practicality of such a system is questionable. Another major drawback is the lack of explainability. Although it is possible to understand the reasoning behind the user's decision to judge an output as right or wrong, the functionality of the chatbot itself in arriving upon the output is unknown and is not determinable. This creates difficulties in debugging issues because when a chatbot fails or performs poorly, it is essential that developers are informed about the specific areas which need to be addressed. However, that is not possible through user feedback.

3. Algorithm Inspection Techniques

Algorithm inspection approach works by analyzing the inner working of the chatbot and ensuring that each individual algorithm is functioning as expected. This helps developers to not only assess the performance of the chatbot but also identify specifically where faults lie and where adjustments are required. This approach overcomes the challenges faced by the output testing approach by being rigorous, intensive and precise.

Among other things, some of the most popular algorithms used by conventional Chatbots are Naïve Bayes, Decision Trees, Support Vector Machines, Recurrent Neural Networks (RNN), Markov Chains, Long Short Term Memory (LSTM) and Natural Language Processing (NLP). Classification algorithms are used by chatbots primarily to identify the intent in phrases. This helps in deriving context pertaining to that intended by the user in the input. Topic Modelling using algorithms such as LDA and feature extraction (NLP) is useful in extracting the overall topic or domain of the conversation. Naïve Bayes is useful for determining how confident that chatbot is in its prediction, and Decision Trees are more appropriate for determining how the chatbot arrived at the classification prediction that it did. RNN and LSTMs are excellent algorithms to process textual data efficiently.

By far the most common and appropriate testing procedure for these algorithms is cross validation. This essentially uses a dataset split into training and test sets to assess how effectively an algorithm has learned to correctly perform its function. Based on this, we can get an effective measure of performance for each part of the chatbot. Cross validation starts by taking a dataset and splitting it into two parts: training and test (typically the split is around 70-30 respectively). An algorithm is trained using the training data and it is evaluated using the test set. Based on the input given to the algorithm it makes predictions and performs intent classification. The number of correct predictions is indicative of the algorithms accuracy and performance. K-fold cross validation is one of the most common cross validation techniques where the model trained on the training set is tested on the test set iteratively for k number of times. The train and test sets are split either using random splits or stratified splits. Through k-fold cross validation, the performance of prediction algorithms can be determined, and adjustments can be made if necessary.

The most significant of these algorithms, and arguable the most important technique within chatbots, is Natural Language Processing (NLP). NLP is responsible for how well a chatbot is able to understand human language, and therefore how well it can generate valid responses. This algorithm must be functioning efficiently if the chatbot is going to have meaningful conversations with the user, which is its primary goal. Although the performance of NLP based Chatbots is subjective, a promising way in which they are often assessed is through some version of the infamous Turing Test. In this, a human converses both with the chatbot and a human, and tries to distinguish between the two. If through the Turing Test a human is unable to identify the chatbot, then we can assume its NLP algorithm is functioning appropriately. Although this is an option, it may not be the most efficient since chatbots are designed to function within specific domains. For example, it would not be desirable to test the performance of a chatbot built for banking domain with questions related to e-commerce. Such scenarios will be outside the intended operating premises of the chatbot. In the subsequent sections, popular chatbot algorithms and their testing methodologies will be explored. Moreover, NLP and its corresponding testing techniques such as verification and validation, grammar and parsing algorithms and statistical parsing are also explored.

3.1. Naïve Bayes Algorithm

Naïve Bayes algorithm attempts to classify text into certain categories so that the chatbot can identify the intent of the user, and thereby narrowing down the possible range of responses. Since intent identification is one of the first and foremost steps in chatbot conversations, it is imperative that this algorithm works properly. The algorithm relies on commonality, which essentially means that certain words should have more weight for particular categories based on the frequency of their appearances in that category [9].

The most straightforward way of testing this algorithm is by using k-fold cross validation. This involves training the chatbot with certain inputs and their corresponding classifications, then using a test set to assess how often the chatbot can correctly classify a given input. We can use confusion matrices, accuracy, precision and recall for assessing the performance of the algorithm.

An issue with Naïve Bayes algorithm is that it uses a 'bag of words' approach. Essentially, the algorithm considers the words as an entire set and selects the most important ones to determine the class of input. This means it does not consider the order of words in which they appear. This may be problematic, as certain rearrangements of words could cause inputs and their classes to differ. To overcome this, techniques such as n-gram can be used to preserve the order of the words. Moreover, Explainable AI (XAI) [6] is an emerging technique that is used to 'explain' machine learning and deep learning models and understand the rationale behind their decisions [7].

3.2. Support Vector Machines

SVMs work based on the principle of Structural Risk Minimization Principle. SVMs work very well with text data and Chatbots because of the high dimensional input space due to large number of text features, linearly separable data and the prominence of sparse matrix. It is one of the most popularly used algorithms for text classification and intent identification.

This allows us to assess how likely an input is to being in one category or another. Cross validation is the most common way of testing this algorithm, assessing how accurate the output produced from this procedure are based on training and test sets. Precision and recall metrics are also used to assess the performance of this model. Based on experiments, it is noted that SVMs consistently outperform similar algorithms such as K-NNs and Naïve Bayes [8].

3.3. Deep Neural Networks

Inspired by the human brain, neural networks consist of layers of interconnected artificial neurons which communicate with one another. These neurons learn features from the data and work together to produce a meaningful output. Neural networks are data intensive and require huge volumes of data to learn patterns and trends in the data. To test this algorithm, we should determine whether the chatbot generates valid response to inputs, keeps the conversation flowing, addresses the need of the user and whether the chatbot is able to mimic the linguistic characteristics of a human to a reasonable extent. This may mean an adaptation of the Turing test may be an appropriate testing method.

The issue with neural networks is the lack of explainability. It is not straightforward to determine which neuron in the network contributed to the prediction or which neuron processed a specific feature. Moreover, neural networks tend to rely on the availability of enormous amount of data so that it can iteratively go through a sufficient learning process and ensure that its responses will be as valid as possible. Such large amount of data is generally not found in chatbot environments. Therefore, it has led to the use of efficient algorithms such as LSTMs and RNNs that work well with textual data.

3.4. Markov Chains

Markov chains find extensive usage in text generation and Chatbots. They work by determining the probability of going from one state to another state [12]. This model is simple to use and summarize as it can be easily stored as matrices. These chains do not take into account the path taken to attain a certain state and rather, depend on the

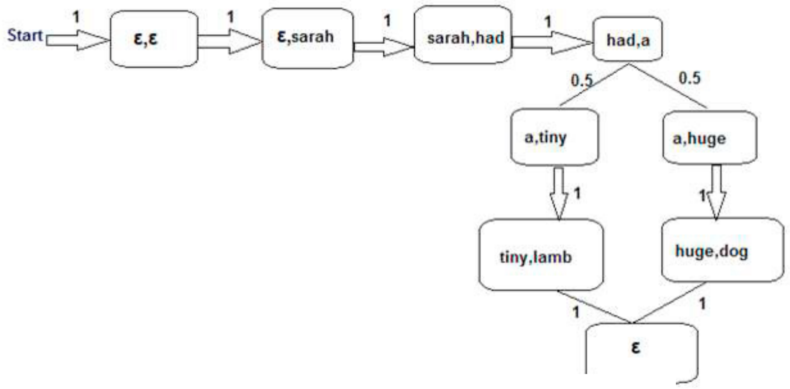

Figure 1: A two order Markov chain for text generation

previous state to determine the current state. In the context of Chatbots, Markov chains work by defining order of chains as depicted in figure 1 (which is a two-order chain). The order of chains refers to the number of words that will be grouped together in each chain with first order models having one word while third order models having a group of three words. Consequently, higher order chains more closely represent the training data and have less variance while low order chains are more random and produce variable output.

The performance of Markov chain based Chatbots can be tested using grammatical parsing, output analysis and user feedback testing. Markov chains work by building responses that are statistically more viable and realistic [13]. However, due to the probability and randomness involved when combining different Markov chains, there could be situations where the output might not meaningful. Such situations must be identified and the algorithm retrained to ensure the chatbot does not generate illegible outputs.

4. Natural Language Processing (NLP)

NLP is of particular importance for chatbots because this technique determines how the bot will understand and interpret the text input. The goal of an ideal chatbot would be to converse with the user in such a way that the user is completely unaware that they are talking with a machine. This algorithm attempts to learn through machine learning and an abundance of conversational data, the intricacies of human language. NLP helps the bot understand text data, comprehend grammar, sentiment and intent. This is primarily due to the wide range of functionalities offered by NLP such as text summarizations, word vectorization, topic modelling, PoS tagging, n-gram and sentiment polarity analysis [11].

Due to the nature of NLP techniques to try and mimic human conversation, testing this algorithm is essentially testing the communication abilities of the chatbot. Testing NLP algorithm is very subjective and there is no absolute standard or benchmark to assess the algorithm's performance. Some of the most popular and appropriate ways to test NLP algorithms are user satisfaction and feedback analysis.

Conducting data analysis and text analysis on user satisfaction ratings may be simple but they do not provide enough insight into the results and are therefore not recommended. Variations of Turing test [5] can be used to test the algorithm as NLP tends to try and mimic human linguistic conversations. The Turing test can be used to assess how close the algorithm is to mimicking actual human interactions and human language. The test could also involve users trying to differentiate between conversations with that of a human and to that of a NLP based chatbot.

To ensure fail-proof testing, it is imperative that the user poses intentionally difficult or error prone inputs to see how the chatbot performs. Chatbots learn what they need to do in order to deliver valid responses, but a good chatbot should also know what it does not need to know (i.e. it knows how to determine both acceptable and invalid responses and inputs). Since this algorithm essentially relies on the chatbot learning from data, we could potentially determine a way to test how appropriate the data is. If the data is not particularly useful, the chatbot will learn from this and not deliver valid responses. Another method of testing here could be to evaluate if the algorithm is able to detect grammatically correct sentences. That is, give the algorithm an input arranged in different ways- jumble the order of words, make it contain grammatical/spelling/punctuation errors etc. and determine if it can identify which sentences are without error.

4.1. Verification and Validation

Another possibility arises when we consider verification and validation (V&V). Validation is used to check whether the algorithm is appropriate for the given requirement. The Turing test is a form of NLP validation used to test AI Chatbots. Verification is used to ensure that the algorithm is built properly, and it works to fulfill its intended goal [3]. It can be performed without running the function or executing the code. 'White box testing' is a form of verification that involves creating test cases then going through the code to determine expected outputs. If these outputs seem reasonable, then 'black box testing' is conducted in which the code is executed and the expected and actual results are compared [4]. For NLP, we could evaluate whether the chatbot can understand human inputs of different forms (i.e. different syntaxes, words, sentence structures etc.) and assess whether the algorithm can handle the various situations presented to it. This process can be used to effectively test most NLP based algorithm.

4.2. Grammar and Parsing Algorithms

To tackle ambiguity, it is important to identify how a chatbot can possibly understand and identify the presence of ambiguity in a sentence. If the system is able to identify the ambiguity pre-emptively, it can then resolve the ambiguity and the conversation can continue to flow. Grammar and parsing algorithms are viable approaches to ambiguity detection and resolution. The grammar algorithm provides a formal description of the structure of a language to ensure that the chatbot communicates without linguistic errors. The parsing algorithm relies on the grammar to analyze the grammatical structure of a sentence, for example,

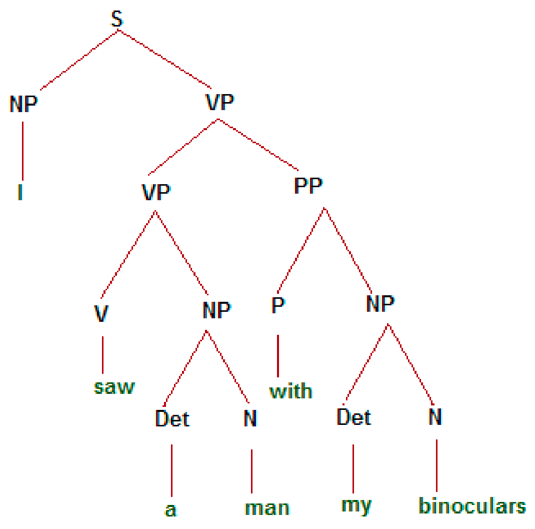

Figure 2: Example of an ambiguous statement

identifying noun phrases, verb phrases etc. these two algorithms are an integral part of AI Chatbots.

Ambiguity causes the grammar functions to be able to parse a sentence in multiple different ways based on the different interpretations. Through white box testing, test cases with ambiguous phrases can be used to determine whether the parser will indeed identify multiple interpretations of the input. Ideally, the chatbot will be trained such that when such a situation is encountered, it asks the user to rephrase the input since it will not be able to comprehend the request. Black box testing can be used after this approach to determine whether the expected output after an ambiguous input actually occurs.

Ambiguity occurs when a sentence can have multiple interpretations as a result of its wording or syntax. As depicted in figure 2, the phrase 'I saw a man with my binoculars' can be interpreted in two entirely different ways – did the binoculars help me see the man or did I see the man in possession of my binoculars. . When a chatbot receives such an ambiguous input, it will not be able to generate a valid response for the user, as it is not trained to do so. This leads to an undesirable state where the chatbot fails and conversation breaks down.

4.3. Statistical Parsing

Statistical parsing is another ambiguity detection technique which works by providing a probability to each parse such that the one with the highest probability is the grammatically correct interpretation. On a well-trained chatbot working within its intended environment, this probability is high but when ambiguity is present, this probability will decrease and there may be a much narrower range of probabilities. The probabilities of ambiguous inputs can be studied and analyzed to determine whether ambiguous test cases generally fall below a certain threshold. If this is the case, then the chatbot can again be trained to identify that the input is ambiguous and can therefore act appropriately upon this information. This enables the chatbot to be trained to recognize why it fails to understand an input. This forms a feedback mechanism that further reinforces the functionality of the bot. If the bot is still not able to comprehend a phrase, it can ask the user to rephrase the sentence in a legible way to ensure optimum functionality.

To completely test the functionality of a chatbot across different key metrics, an ensemble or staged-approach can be used where the discussed testing techniques can be used together. This would help strengthen the performance of the chatbot as it is tested and evaluated through a variety of techniques and scenarios. Moreover, the Chatbots can learn from examples of ambiguous situations presented to them to efficiently handle similar scenarios in the future.

5. Conclusion

With Chatbots seeing widespread usage across domains and various sectors, it is imperative that they are thoroughly tested and validated. Although a variety of techniques exist to test Chatbots such as output testing, algorithm testing is one of the most promising. This is especially due to the fact that it is difficult to inspect and analyze the performance of NLP based techniques often incorporated in Chatbots. Moreover, ambiguous statements have always posed challenges to the functionality of Chatbots with most conventional bots failing or unable to respond meaningfully. To overcome this, verification and validation approaches coupled with statistical parsing and other ambiguity detection tools will be key. This ensures that when such an ambiguous or irrelevant input is detected, the bot asks the user to rephrase the question appropriately. Such implementations will make interactions with Chatbots of the future indistinguishable from human interactions.

References

1. Nicole Radziwill and Morgan Benton (2017), Evaluating Quality of Chatbots and Intelligent Conversational Agents.
2. https://chatbottest.com/, Accessed June 2019.
3. Andrei.Popescu-Belis, (1999), Evaluation of natural language processing systems: a model for coherence verification of quality measures.
4. Valerie B. Barr, Judith L. Klavans, (2001), Verification and Validation of Language Processing Systems: Is It Evaluation?
5. A. M. Turing (1950), Computing Machinery and Intelligence. Mind 49: 433-460

6. Wojciech Samek, Thomas Wiegand, Klaus-Robert Müller, (2017), Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models.
7. Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin, (2016), "Why Should I Trust You?": Explaining the Predictions of Any Classifier.
8. Thorsten Joachims (1998), Text Categorization with Support Vector Machines: Learning with Many Relevant Features.
9. Andrew McCallum, Kamal Nigam, (1998), A Comparison of Event Models for Naive Bayes Text Classification.
10. Sophia Brooke, (2017), Chatbot Testing: Specifics and Techniques, Accessed May 2019.
11. Sameera A. Abdul-Kader, Dr. John Woods (2015), Survey on Chatbot Design Techniques in Speech Conversation Systems, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 6, No. 7.
12. Cameron Franz, (2014), A Simple Markov Chain Chatbot, https://cameron.cf/posts/2014-04-13-Markov%20Chain%20Chatbot.html, Accessed May 2019.
13. Luka Bradeško, Dunja Mladenić (2012), A Survey of Chabot Systems through a Loebner Prize Competition.