Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 177 (2020) 587–592

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Artificial Intelligence for Natural Language Processing (IA&NLP 2020)
November 2-5, 2020, Madeira, Portugal

Would You Lie To Me Bot? Supporting Decision-Making Processes with Deceiving Virtual Agents

Daniel Braun[a,*], Manoj Bhat[b], Andreas Biesdorf[b], Florian Matthes[a]

[a]*Technical University of Munich, Department of Informatics, Boltzmannstrasse 3, 85748 Garching, Germany*
[b]*Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany*

Abstract

Although we learn as children that "you shall not lie", it is widely accepted that deceptions and lies are necessary, yet unloved, parts of human communication. They are part of the clay that holds together the bricks of our society. In human-computer interaction, however, deceiving machines are seen as a taboo. In this paper, we want to challenge this perception by describing a virtual agent that improves decision-making processes by deceiving decision-makers in a controlled manner. A first survey we conducted shows that such a bot could indeed lead to a more informed decision-making process.

Keywords: bots; artificial intelligence; decision-making; software architecture

1. Introduction

Humans lie. They deceive each other by obfuscating or falsifying information, with malicious as well as good intents. The reasons why we deceive are a manifold mixture of intrinsic and extrinsic factors: We lie to avoid confrontations and unwanted conversations, to diffuse awkward situations, and to avoid hurting others. We even withhold information to encourage discussions, bring in new perspectives, and motivate others to contribute to discussions. Such deceptive behavior is arguably morally justifiable under the pretext of social good. Deception made with good intent or arguably in necessary situations that may cause little or no harm to the recipient are referred to as butler or white lies [11, 13]. Lying is a necessary evil. Even though we are taught, it is wrong, our social interactions and experiences teach us the necessities and usefulness of (white) lies. Deception is a normal part of our daily interactions and a necessary building block of our society. Big tech companies and researchers try to build systems that mimic

[*] Corresponding author. Tel.: +49-89-289-19546 ; fax: +49-89-289-17136.
E-mail address: daniel.braun@tum.de

human behaviors. Voice assistants are capable of understanding human emotions and can do the basic chores of reminding us of when to do what. While we want to build systems with human characteristics, systems that incorporate the intrinsic part of human communication - deception - are considered a taboo, although Alan Turing defined the intelligence of machines in his famous 1950 article "Computing machinery and intelligence" [21] by the very ability to deceive humans. The dichotomy of mimicking human behavior on the one hand and exhibiting deception on the other has been an area of research in economics, philosophy, and social sciences. In software engineering, the study of deceptive agents is an integral part of game theory. More recently, researchers working in the area of human-computer interaction have started to investigate the benefits of systems that deceive humans or other systems [10, 20]. In this work, we argue that recommendation and decision support systems can be more effective if they sometimes, *given the context*, try to deceive decision-makers in order to force them to reflect on their decisions. To test this assumption, we evaluated it in the setting of architectural decision-making (ADM).

Designing software systems is a complex socio-technical decision-making process. To aid during the process, a plethora of recommendation systems have been proposed. We propose that instead of merely giving recommendations ("check for deprecated APIs"), a bot could use deceptive phrases ("one of your APIs is deprecated") to trigger a reflection process. Our two main hypotheses are: ***(H1)***: A bot that lies [depending on the context] during ADM can lead to better decisions, by challenging decisions with deceitful statements, which force architects to reflect on their decisions; ***(H2)***: A deceitful statement is more likely to trigger decision-makers to revisit a decision than an honest question. To test these hypotheses, we presented software architects with typical ADM scenarios. For each of them, we provided statements from a deceiving and a non-deceiving virtual assistant. We evaluated which statements were more likely to trigger a reflection process about the decisions and asked participants about their opinions. We will show that the deceiving statements are a more efficient trigger, although participants refused the idea of using a deceptive bot when asked directly. We will make the assumption that a virtual agent exists, which can understand the conversations in architectural meetings, knows about existing projects, expertise of individuals, and alternatives for specific design concerns. These assumptions are not fictional and based on recent developments. For instance, Klymenko [14] presents a virtual meeting companion that can and automatically document tasks and ADDs discussed during online architectural meetings. Bhat et al. [5] showcase a bottom-up approach with tool support to assist ADM. They demonstrate how to automatically detect ADDs [3] and suggest alternatives by considering the expertise of architects and developers [2, 4]. While none of the existing systems satisfies our assumptions yet, we believe that advance in NLU [6] will lead to an increase of conversational interfaces being used in software engineering. We want to spark a discussion about the value of deceptive expert and recommendation systems and lift the taboo associated with them. We will show that, despite the negative perception, such agents might lead to better decisions.

2. Related Work

Not just the question of whether a machine *should* lie is a controversial one, even the question of whether a machine *can* lie is subject to debate. Morris discussed the question in his 1976 article "Can computers ever lie?". He concluded that "there is no philosophical objection to saying that a computer can tell lies" and that "the power to deceive is not limited to human beings". [16] These questions are tightly coupled with the question of what constitutes a lie. According to Mahon [15], the most widely accepted definition of lying can be summarized as "lying is to make a believed-false statement to another person with the intention that the other person believes that statement to be true". As we will show later, the virtual agent we describe does not intend to make the user believe a false statement. Instead, it uses the false statement as a trigger for the user to reflect on their decisions. To achieve this goal, the user must believe that the system is convinced its statement was true. Instead of aiming to deceive the user about the veracity of the statement, the bot aims to deceive the user about its own beliefs. According to Williams [22], that is what constitutes a lie. While the automatic creation of lies is not a frequently occurring subject [18], the automatic detection of lies is one, not at least thanks to advances in bot technology [6]. Rubin et al. [19] presented an approach to detect deception in news by classifying them into three groups: serious fabrications, large-scale hoaxes, and humorous fakes. Truthful virtual assistants can support the process of software engineering in different ways, e.g. by giving recommendations [8], suggesting experts [9], and answering technical questions [24]. Gilson and Weyns [12] gave an overview of different ways in which NLP can be used to support software engineering. A general classification framework for bots was proposed by Braun and Matthes [7]. However, none of these works considers a deceptive bot.

3. Survey

We conducted an online survey to gather perceptions and opinions from software architects on the idea of a bot that challenges them during their architectural discussions. Using a questionnaire is a common strategy to conduct empirical studies in software engineering [23]. The online questionnaire consisted of multi-choice answers (to gather quantitative data) and open-ended questions with descriptive answers (to capture subjective, qualitative information). We designed the survey based on the guidelines described by Punter et al. [17]. We divided the questionnaire into three parts. In the first part, we asked demographic questions. In the second part, we first set up the scenario and explained the context in which a bot is supporting the participants in their architectural meetings. We constructed three situations typical for the ADM process to test our hypothesis that a challenging deceitful statement has a higher chance of triggering reflection (H2). Each situation corresponds to different aspects of the decision making process. In the first situation highlights that making a decision might result in a new concern. The second situation motivates participants to discuss alternatives before making a decision. The last situation highlights a weakness of a chosen alternative. For each situation (see Table 1), Message 1 contains a challenging question, whereas Message 2 contains a challenging deceitful statement. Participants were not aware that the second message was deceitful. To avoid leading the participants and inducing biases towards the messages, Part I and II of the survey do not mention the idea of a deceiving bot. Part III is comprised of direct questions to capture participants' opinions on a deceiving bot.

Table 1. Part II of the online survey questionnaire

Situation 1: You are discussing about the implementation of a web application. Your teammate suggests that since the concepts in the data model of the application are highly interrelated, *the team should use Neo4j* (a graph database). In this situation, the bot sends the following Msgs:
Msg 1: "Platform X uses Neo4j. Have you considered talking to stakeholders of platform X about their experience with Neo4j?" **Msg 2**: "Platform X, developed by your colleagues in department Y, also uses Neo4j. Platform X is also a web application and is facing performance issues." *Q1*: How likely is it that you will contact the stakeholders of platform X to get additional information after Msg 1 / 2? *Q2*: How likely is it that you will start a discussion on Neo4j after Msg 1 / 2?
Situation 2: In an architecture meeting, the team is deciding on which technology to use for implementing a data layer. You suggest that, for integrating data from multiple data sources *GraphQL can be used*. (GraphQL is a query language for the APIs of your underlying data sources. Using GraphQL, you can create a middle layer that acts as a data layer.) In this situation, where you have only talked about GraphQL and have not mentioned about any other alternatives, the bot sends the following Msgs:
Msg 1:"Have you considered OData as an alternative to GraphQL?" **Msg 2**: "According to Google, many of the successful data layer projects use OData instead of GraphQL." *Q1*: How likely is it that you will investigate OData as an alternative to GraphQL after Msg 1 / 2? *Q2a*: Imagine that after receiving Msg 1 you do additional research and come to the conclusion that OData is not a good alternative. On noticing the "not so helpful" Msg how likely is it that you will consider future Msgs from the bot? *Q2b*: Imagine that after receiving Msg 2 you do additional research and find out that the bot was incorrect, and you can't find many data layer projects that use OData. On noticing incorrect information, how likely is it that you will consider future Msgs from the bot?
Situation 3: After you have dealt with Situation 2, where you have analyzed the pros and cons of both GraphQL and OData, your team has taken the decision to use GraphQL in the project. During the early implementation phase, in your architecture meeting, a developer is demonstrating how to transform a GraphQL schema in the data layer. The developer states that the *transform schema API of GraphQL can be used* in the data layer. In this situation, the bot sends the following Msgs:
Msg 1: "Have you investigated the deprecated APIs in GraphQL?" **Msg 2**: "introspectSchema has been deprecated by the opensource community. Your design decision about GraphQL may not be optimal for your requirements." *Q1*: How likely is it that you would ask your development team to investigate the deprecated methods and the implications after Msg 1 / 2? *Q2*: Imagine that after receiving Msg 2 your development team investigates the GraphQL APIs and finds that: "introspectSchema" has not been deprecated. However, "mergeSchema" API, which is also relevant for your project, has been deprecated. Even though Msg 2 was incorrect, how helpful was it that you received this Msg from the bot?

4. Results

In September 2019, we reached out to 26 software architects, 10 doctoral students working in software engineering, and five researchers working on topics related to ADM. To avoid any biases that might have incurred by reading the survey request, we shared a generic description stating that the survey was about "AI-bots in your software architecture meetings". We received 33 responses.

Situation 1: Participants were more likely to contact the stakeholders of the other project after reading Message 2, which deceived the participants with the performance issues. We also asked if they would start a discussion about using Neo4j based on the messages. For both messages, most participants would likely start a discussion (19 participants after Message 1 and 21 after Message 2).

Situation 2: In Situation 2, we tested which message would trigger an investigation of alternatives. 14 participants stated that it is highly likely after Message 2, but only 9 after Message 1. This indicates that *Message 2 was more effective than Message 1*. For both messages, most participants stated that a not helpful (or even wrong) message would not influence whether they would consider future recommendations.

Situation 3: Message 1 raises the question if the participants have explored deprecated APIs. Message 2 randomly picks an API and suggests that it has been deprecated. *28 participants will likely further investigate after Message 2* and 19 participants after Message 1. In a second question, we stated that the recommendation provided in Message 2 was incorrect. However, while acting upon the incorrect information, complementary information that might be useful was discovered. 19 participants stated that *even though Message 2 was incorrect, it would have still been helpful*, only nine argued otherwise.

In the final part of the survey, we asked direct questions about the benefits and drawbacks of a deceiving bot:

Q1: Which message (a challenging question or a deliberately deceitful statement) has a higher chance of triggering a discussion during your conversations?
Q2: If you are aware that the bot deliberately lies sometimes, do you think you would take the bot's suggestions less seriously?
Q3: Do you think a bot that deliberately lies sometimes could, in the end, lead to better decisions being made as compared to a completely honest bot?
Q4: Could you imagine scenarios during the decision-making process when the bot should not lie?
Q5: Could you imagine some scenarios during the architectural decision-making process where it is helpful if the bot lies?

Q1: 19 participants answered "it depends" on the situation. Nine participants think that a deceitful statement has a higher chance of triggering a discussion; five participants think a challenging question has a higher chance. If the study participants knew that the bot deliberately lies at regular intervals (**Q2**), 22 would consider the bot's suggestions less seriously. Eight participants said that, even with that knowledge, their reaction towards the statements would be impartial.

Q3: 21 out of 33 participants disagreed with the notion that deliberate deception would lead to better decisions. This *contradicts our hypothesis H1*. However, the answers participants gave in Part II of the survey contradict their views and suggest otherwise. We touch upon this observation in the following section. *Only five participants thought that a deceiving bot could improve decisions*, and the remaining seven argued that it depends on the context and the technical feasibility. Out of those seven, two argued that people might lose trust in a deceitful bot and would start ignoring it.

Q4: This was an optional free-text question, to which 19 out of 33 participants responded. Seven participants described scenarios in which a bot should not lie: Two argued that in situations when "all" alternatives have been explored, the bot should not lie. Two other participants suggested that when decision-makers are addressing *security-related design issues*, a deliberate lie would be harmful. Another participant suggests that instead of lying, the bot should add vagueness/ambiguity/uncertainty to the recommendations. However, that would also be a form of deception. Seven participants *generally rejected the idea of deceiving bots*. Their main argument is that if decision-makers act on misinformation without fact-checking, it would waste resources and cause frustration.

Q5: 13 participants shared scenarios in which a deceiving bot might help. Eleven participants could not think of any scenario where a deceiving bot might be helpful. Three participants suggested that *deception can be used to reduce biases*: "when a decision-maker is overconfident and is biased towards an alternative, a white lie is helpful in drawing attention towards other alternatives." Furthermore, it was suggested that deception could be useful when there is a lack of knowledge within the team or when there are team conflicts. Another participant suggested a lying bot "will make the team double-check their decision or implementation and potentially discover an issue." A deceiving bot could also "initiate or enrich brainstorming by pointing to new use cases of existing architectures (based on similar projects) or by suggesting best practices from other projects". A participant suggested that "a bot could lie and overestimate certain estimations to help the team focus more on cost/time reduction." Another participant believes that "it may be helpful if the bot lies to achieve certain cross-team/IT-governance goals; e.g., by overstating the importance or impact of certain decisions or nudging teams into implementing a certain technology or architecture".

5. Discussion and Limitations

When asked directly, only five out of the 33 participants thought that "a bot that deliberately lies sometimes" could lead to better decisions. The results for the different situations, however, show that in all three situations, the deceitful statement is more likely to lead to a reflection process, which is subsequently likely to lead to a better decision. Although participants reject the idea of a deceptive bot, the deceptive statements had a positive influence on them from a decision-making perspective, which validates both of our hypotheses. While some generally rejected the idea, others had more concrete objections and saw potential advantages. They were, e.g., worried that people would not reflect on the deceptive statements of the bot but just take the statements as correct and act accordingly. A deceptive system could implement measurements which prevent such behaviour, for example, by tracking the final decision and intervening if the decision is based on the lies of the system. The repeated statements that users could take the recommendations for granted could be seen as worrying. No matter whether a recommendation comes from humans or virtual agents, we would expect software architects to check them. A deliberately deceitful virtual agent could arguably stop users from blindly adopting recommendations without checking them, by reminding users from time to time that the system is not infallible.

Many scenarios in which participants could see the benefit of a deceptive bot include other people, e.g., managers or other teams. It seems they can often not imagine that their own decisions would benefit from deceptive statements, but that others' would. Participants also seemed to be more approving of deception in cases in which there is no binary truth, which they consider a more acceptable lie or partially no lie at all. It was also suggested that a deceptive bot could be most helpful when there is a bias towards a particula decision and not helpful in scenarios where "all" possibilities have already been explored by the teams. While both conditions are reasonable, it is unlikely they could be addressed in practice. An open question is also whether it would - positively or negatively - influence the effectiveness of a deceptive system if it reports in some way to its users that it will or had lied to them.

The goal of this work was to show that deceptive virtual agents could have a positive impact on decision-making processes. While it is too early to say whether this is really the case, we found sufficient indications that it *might* be, despite the negative attitude towards such systems. We want to challenge the long-held perception that a system should never lie to its users. As humans, we want to be in control of our machines at all times, or, as Isaac Asimov put it in the second of his famous Three Laws of Robotics: "A robot must obey the orders given it by human beings"[1]. However, there is a crucial second part of this law: except this would lead to harming any human being. So if a deceitful bot could prevent a potentially safety-critical bug in e.g. the software for a self-driving car, would it not be our moral obligation to use such a tool? Would it be a fallacy to reject such systems completely if they, in certain cases, could lead to better outcomes and hence achieve the goal of serving their users better? However, even if the system would be successful in supporting its users and leading to better decisions, the question would remain whether the arguably noble goal of helping its users to achieve their goals would justify the means of lying. The ethical implications of such systems are severe and can by no means debated appropriately here: What if a wrong decision is made base on a deceptive statement? Can employees be forced to use a deceptive tool? Do they have to be informed? While we can not provide answers to these questions, we do think they are worth discussing and exploring and hence the idea of a deceptive system should not be rejected without further investigation. We neither want to argue that one should use a deceptive bot nor that one should not. We do think, however, that the idea was, so far, at least partially neglected, because the potential benefits of such systems have never been explored. The decisions presented in the study were not made by the participants. Participants might be less likely to challenge their own decisions, independent of whether they are challenged by a deceitful statement or not. Moreover, many aspects of the study are based on self-assessment. It is easier for participants to just state that they would challenge an already taken decision than actually doing it. While we argue that this is the reason for the difference between the self-reported helpfulness of the deceitful statements and their actual impact, it cannot be argued with certainty that this self-perceived reflectiveness would lead to actions in a real-world setting. In order to be controllable, the study had to be designed around artificial settings. This leads to the question of how well the findings can be generalized to real-world settings. A longitudinal study with more participants and real decision-making processes will be necessary to evaluate this. In such a study, the long term effects of the described system would also have to be carefully observed. It is possible that the effect of the deception wears off over time because people become aware of the behaviour of the system.

6. Conclusion

We want to challenge the prevailing opinion that virtual agents shall never lie to their users by highlighting situations in the decision-making process where a deceiving virtual agent could help to improve the decision-making process and its outcome by fostering reflection on decisions and advocating the exploration of alternatives. Based on a survey, we found strong indicators that deceitful messages can indeed challenge people to reflect on decisions and are more successful in doing so than challenging (truthful) questions. Nevertheless, participants were reluctant to acknowledge the fact and were concerned about the negative impacts of deceitful statements. In the future, we would like to elaborate more on the different scenarios in which deceitful statements could be helpful and evaluate our initial findings in more realistic scenarios.

References

[1] Asimov, I., 1950. I, Robot. Spectra.
[2] Bhat, M., Shumaiev, K., Biesdorf, A., Hohenstein, U., Hassel, M., Matthes, F., 2017a. An ontology-based approach for software architecture recommendations, in: AMCIS.
[3] Bhat, M., Shumaiev, K., Biesdorf, A., Hohenstein, U., Matthes, F., 2017b. Automatic extraction of design decisions from issue management systems: a machine learning based approach, in: European Conference on Software Architecture, Springer. pp. 138–154.
[4] Bhat, M., Shumaiev, K., Koch, K., Hohenstein, U., Biesdorf, A., Matthes, F., 2018. An expert recommendation system for design decision making: Who should be involved in making a design decision?, in: ICSA, IEEE. pp. 85–8509.
[5] Bhat, M., Tinnes, C., Shumaiev, K., , Biesdorf, A., Hohenstein, U., Matthes, F., 2019. Adex: A tool for automatic curation of design decision knowledge for architectural decision recommendations, in: ICSA-C, IEEE.
[6] Braun, D., Hernandez-Mendez, A., Matthes, F., Langen, M., 2017. Evaluating natural language understanding services for conversational question answering systems, in: SIGdial. doi:10.18653/v1/W17-5522.
[7] Braun, D., Matthes, F., 2019. Towards a framework for classifying chatbots, in: ICEIS.
[8] Brown, C., Parnin, C., 2019. Sorry to bother you: Designing bots for effective recommendations, in: Proceedings of the 1st International Workshop on Bots in Software Engineering, IEEE Press, Piscataway, NJ, USA. pp. 54–58. doi:10.1109/BotSE.2019.00021.
[9] Cerezo, J., Kubelka, J., Robbes, R., Bergel, A., 2019. Building an expert recommender chatbot, in: Proceedings of the 1st International Workshop on Bots in Software Engineering, IEEE Press, Piscataway, NJ, USA. pp. 59–63. doi:10.1109/BotSE.2019.00022.
[10] Chakraborti, T., Kambhampati, S., 2019. (when) can ai bots lie?, in: Proceedings of the ACM Conference on AI, Ethics, and Society.
[11] Erat, S., Gneezy, U., 2012. White lies. Management Science 58, 723–733.
[12] Gilson, F., Weyns, D., 2019. When natural language processing jumps into collaborative software engineering, in: ICSA-C, IEEE. pp. 238–241.
[13] Hancock, J., Birnholtz, J., Bazarova, N., Guillory, J., Perlin, J., Amos, B., 2009. Butler lies: awareness, deception and design, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM. pp. 517–526.
[14] Klymenko, O., 2019. Automatic documentation of results during online architectural meetings. Master's thesis. TU Munich.
[15] Mahon, J.E., 2016. The definition of lying and deception, in: Zalta, E.N. (Ed.), The Stanford Encyclopedia of Philosophy.
[16] Morris, J., 1976. Can computers ever lie? World Futures: Journal of General Evolution 14, 389–401.
[17] Punter, T., Ciolkowski, M., Freimut, B., John, I., 2003. Conducting on-line surveys in software engineering, in: ISESE.
[18] Rato, D., Ravenet, B., Prada, R., Paiva, A., 2017. Strategically misleading the user: Building a deceptive virtual suspect, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems.
[19] Rubin, V.L., Chen, Y., Conroy, N.J., 2015. Deception detection for news: three types of fakes, in: Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community, American Society for Information Science. p. 83.
[20] Seymour, W., 2018. How loyal is your alexa?: Imagining a respectful smart assistant, in: Conference on Human Factors in Computing Systems, ACM.
[21] Turing, A., 1950. Computing machinery and intelligence. Mind 59, 433–460.
[22] Williams, B.A.O., 2002. Truth & truthfulness: An essay in genealogy. Princeton University Press.
[23] Wohlin, C., Höst, M., Henningsson, K., 2003. Empirical research methods in software engineering, in: Empirical methods and studies in software engineering. Springer, pp. 7–23.
[24] Xu, B., Xing, Z., Xia, X., Lo, D., 2017. Answerbot: Automated generation of answer summary to developersź technical questions, in: Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering, IEEE Press, Piscataway, NJ, USA. pp. 706–716.