Pescetelli, N., Cebrian, M., & Rahwan, I. (2020). BeeMe: Real-time internet control of situated human agents. *Computer, 53*(8), 49-58. https://dx.doi.org/10.1109/MC.2020.2996824

This paper was published by IEEE as:
Pescetelli, N., Cebrian, M., & Rahwan, I. (2020). BeeMe: Real-time internet control of situated human agents. *Computer, 53*(8), 49-58. https://dx.doi.org/10.1109/MC.2020.2996824.

BeeMe: Real-Time Internet Control of Situated Human Agents

Niccolo Pescetelli, Manuel Cebrian, and Iyad Rahwan,
Max Planck Institute for Human Development

We present BeeMe, an online platform designed for Internet collective action and problem solving. As a test case, we analyze data from a global performance where thousands of individuals collectively solved a mystery online. We discuss our results in light of contemporary debates on hybrid systems.

In this article, we describe BeeMe, an open platform that we created to experiment with creative solutions to scalable collective action. The platform is accessible at https://beeme.online. BeeMe is the first attempt to build a real-time collective-decision system for open-ended tasks. It allows multiple users to observe and collectively control the actions of one human surrogate (the agent) operating in physical space during scheduled events. The platform was designed as an exploration to scale up humans' abilities as extreme co-operators.

Through evolutionary and cultural innovations— such as language as well as cultural norms and institutions— humans' competitive advantage lies in the ability to cooperate in large-scale societies of mostly unrelated individuals.[1,2] However, the time scale at which such institutions adapt in response to new goals or environments is much larger than that on which individuals operate. Digital communication technologies promise to greatly speed up the time scale on which large collectives can demonstrate adaptive intelligent behavior.[3,4]

One of the main remaining challenges in creating real-time collective-decision systems is how to rapidly aggregate preferences and extract the intentions of a crowd. Even

Originally published in: Computer, 53(8), 2020, p. 49

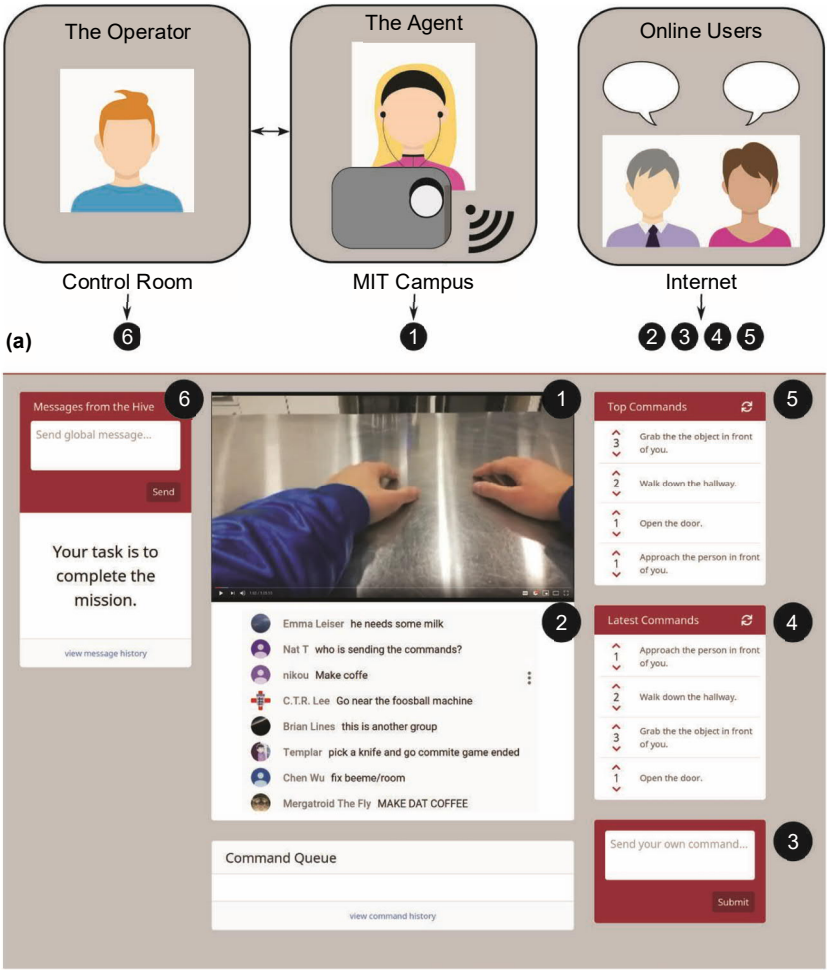

FIGURE 1. The schematic of the setting and BeeMe platform interface. (a) An agent streams his/her surroundings via a camera device. Actions to the agent are communicated by the operator, who monitors the online conversation and outputs new missions to the users. Online users coordinate their actions via the interface. The arrows represent information channels or inputs. (b) (1) A video stream shows the agent's surroundings. (2) A chat messaging system allows online users to coordinate strategies to solve each task. (3) A command box allows users to offer suggestions for possible actions. (4) Users can up-vote or down-vote each others' commands based on the latest suggestions. (5) A running tally of the most popular commands is also shown to users for control actions that are about to be communicated to the agent. (6) An operator box allows the operator to send messages to online users to communicate new missions and advance the narrative.

though machines have achieved superhuman results in many domains of knowledge,[5] whether they can be used for the betterment of our political life and/or the enforcement of social norms remains unclear.[6] This is, partly, because machines still lack the ability for situated problem solving in open-ended tasks; although this field is rapidly progressing, [7,8] adapting decisions to context-specific information in dynamic environments remains challenging for machines. In contrast, humans are good at rapid reasoning and intuitive judgments that take into consideration dynamic and contextual information.

The creation of scalable and flexible decision systems that integrate human, collective, and machine intelligence remains a challenging yet crucial goal for society.[9,10] Substantial prior research has been done in the field of collective decision making, enabling distributed populations to converge on solutions in real time.[11] However, prior studies have used highly controlled tasks within a decision space of defined options.[12,13] BeeMe was designed for testing collective decisions in free-form environments with an open-ended set of actions. Thus, BeeMe allows an Internet crowd to assist situated human agents, namely, agents acting in physical environments that are typically time variant and dynamic and require contextual knowledge.[7,9]

As a first case study, we analyze data obtained during an Internet performance, which went live on Halloween night 2018, when over 1,900 users played simultaneously.[14] Online users had to coordinate their suggestions to control two human avatars, the agents, using an inbuilt chat and/or a rank voting system, to solve a science fiction (sci-fi) mystery. The agents had to be guided through a series of missions that unlocked new tasks and developed the narrative further. Although the narrative gave a structure to the game, users were free to choose whatever action they preferred.[15] Another confederate (the operator) aggregated users' suggestions into single commands and relayed them to the agent (Figure 1). The operator's role was,

Originally published in: Computer, 53(8), 2020, p. 50

thus, to interpret the crowd's intentions and aggregate diverse suggestions into discrete actions. Whether these would have resulted in a coherent goal-directed plan—ultimately, leading to completion of the game mission—or a sequence of disconnected actions was unknown at the time.

The results suggest that human operators were often biased and did not follow the majority. For example, they tended to relay commands that were mentioned only once and, thus, not representative of the crowd's will (called here, for simplicity, *singletons*). For this reason, simple algorithms trained on raw human data had difficulties fitting more than a third of the operators' issued commands. However, after filtering out such arbitrary segments, algorithmic performance greatly improved.

Six algorithms were designed to read human discussions (chat logs) and map them into relevant action commands representing the democratic view of the crowd. We find that machine-generated commands often loosely match human-generated commands, that is, human-generated commands appear among the first five commands ranked by the algorithm. When the human- and machine-generated commands differ, machine-generated actions are still sensible, given the recent discussion. Based on these findings, we created an online aggregator that can be deployed on real-time chat streams to output commands. Such an algorithm may achieve more democratic results in reading the crowd, thus reducing human bias. We discuss these findings in light of the recent debate on artificial intelligence (AI) and governance and examine how they can help in designing hybrid collective-decision systems.

THE PLATFORM

BeeMe was designed to allow a large Internet crowd to remotely control the actions in physical space of one agent to complete simple games (here, a sci-fi mystery; a documentary[21] has been published for more details). The agent streamed the surroundings to the platform using a camera device. The agent only performed actions that were communicated to him/her via an earpiece by another confederate (the operator) monitoring the platform from a control room. The operator was clearly instructed to communicate to the agent only commands that were agreed upon by the crowd.

The online user interface [Figure 1(b)] consists of (1) a central video stream where online users can observe the surroundings of the agent; (2) a chat system that allows users to coordinate and socialize; (3) a command suggestion box, where users can suggest their own commands (such as "turn left" or "run away"); (4) and (5) a voting mechanism that users can use to up-vote and down-vote others' commands, based on (4) recency or (5) popularity; and (6) an operator text box, where the operator could send messages to all users to give the crowd new missions or advance the game's narrative.

Unknown to them, online users were randomly assigned to two teams, each controlling a different character in the story (that is, a different agent) who was freely roaming the Massachusetts Institute of Technology campus. The game consisted of a series of 10 missions that had to be carried out under time pressure to defeat the characters' foe (see the supplementary material for details). The 10 tasks were designed to cover a wide range of behavioral functions of progressive difficulty: motor (tasks 1, 3, and 9), navigation (tasks 2, 4, and 8), social–linguistic (tasks 5 and 6), motor coordination with the other character (task 7), and, finally, a moral dilemma (10).

Although the platform allowed for automatic voting of single action suggestions, users ended up giving their suggestions via the inbuilt chat due to its lower latency. The operators, thus, had to intuitively aggregate the chat logs in real time into a stream of commands to be delivered to the agent. This unexpected mode of interaction allowed us to investigate how an algorithm reading a stream of chat messages could imitate human operators in interpreting the collective will.

HALLOWEEN 2018: A CASE STUDY

The Internet performance involved 1,935 unique logins in the 3 h before and after the event, 683 unique active chat users, and 6,102 unique chat messages (a 2,505/3,597 split for the two crowds). For the purpose of analysis, there were two data streams: the chat log and the commands issued by the operators. The chat log represents a list of chat messages, each with a body, a time stamp, and a username associated with that message. The entries are direct written inputs from users. The commands issued by the human operators were performed by the agent during the live event and are represented by a text body and a time stamp.

We segmented the chat data based on the commands issued by the operators [Figure 2(a)]. The term *segment* denotes the time window between two consecutive actions. The event lasted for about 1 h. The total numbers of actions performed were 91 and 84 for each of the two characters. The small number of data points did not allow the use of sophisticated

Originally published in: Computer, 53(8), 2020, p. 51

natural language processing techniques (for example, using deep learning). Instead, we opted for a more viable rule-based approach, which consisted of reducing all commands and chat messages to noun-plus-verb pairs and performing statistical tests on those. We fit six algorithms to one character's data, selected at random, and used the second character's data as the test set.

Qualitative Observations

Both crowds managed to coordinate their suggestions and to complete all 10 missions that the game entailed. Distributed collective control was possible despite the adversarial effects of communication delays, trolling, and social miscoordination. As judged by the operators, the crowd had an easier time controlling motor and navigation tasks rather than linguistic ones, likely due to the delays in the stream. The game ended with a prisoner dilemma situation where each character (that is, that character's crowd)

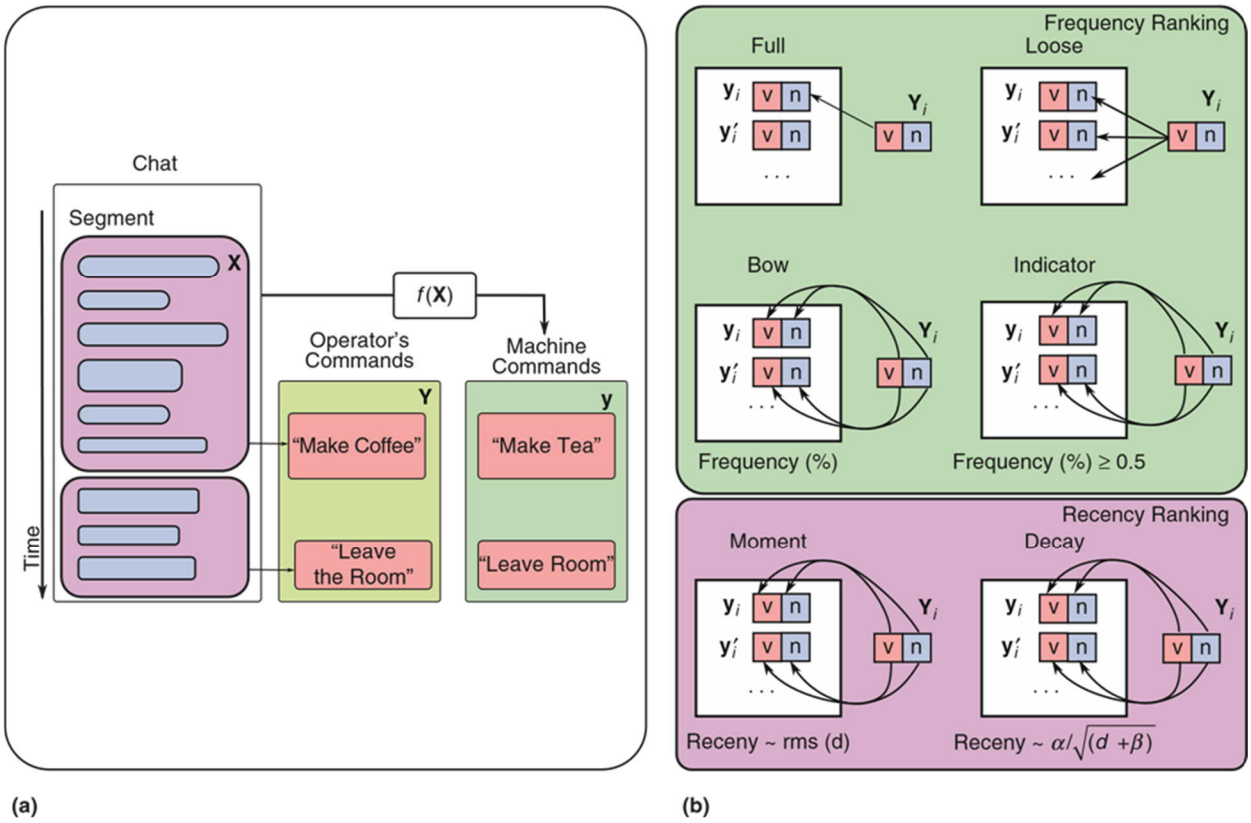

FIGURE 2. (a) The data segmentation and algorithmic task. Chat data were segmented based on the commands issued by the operators to the agents. Each chat segment and the associated operator's command represented one data point. Each chat message within the segment was further processed by mapping it into a relevant bigram (not shown). Six algorithms were designed to read chat logs and output for each segment a (bigram) command. The objective was to maximize the match between human-generated and machine-generated bigrams. (b) A graphical representation of the six algorithms tested. Each algorithm ranks the set of available bigrams generated from the chat based on frequency (green panel) and recency (purple panel). Rank is represented in descending order by superscript (**y, y′**, …). Arrows represent comparisons/matches between operator-issued bigrams **Y** and bigrams obtained from users' suggestions (**y**). Comparisons were either between entire bigrams (first row) or unordered words within bigrams (second and third rows). v: verb; n: noun.

Originally published in: Computer, 53(8), 2020, p. 52

had to decide whether to betray the other. Both crowds opted for cooperation. Second, visible patterns emerged during the experiment. These are worth mentioning as they can help formulate the model and/or suggest questions for further study. The patterns discussed here are as follows: frequent chat commands did not always win, spillover effects, users' assessments of the system's functionality, and user behavioral patterns.

Frequent Suggestions Did Not Always Win. Visual inspect ion showed that, for a large cluster of actions (36% and 22% in the two crowds), the command that the agent received was one of the most frequent commands suggested by the crowd. As we are interested in designing democratic algorithms, we focus on these cases. In the rest of the cases, the operator's commands were biased toward nonrepresentative views, either due to the operator's cognitive overload or a deliberate attempt to advance the story. Although outside the scope of this article, modeling such idiosyncratic operators decisions is highly informative and should be addressed in the future.

Spillover Effects. On top of the frequency of a command, its recency was also an important factor in determining whether a user's suggestion was predictive of the operator's next command issued to the agent. We observed that the initial approximately 40% of each chat segment often referred to the previous (rather than the following) action. We refer to this effect as *user spillover*.

A second form of spillover was *operator spillover*. This happened when, between two similarly popular commands on segment i (for example, "make tea" and "leave the room"), the operator chose one as the action to perform (for example, "make tea"). Then, once that action was performed, the operator transmitted the second one (for example, "leave the room") on segment $i + 1$, disregarding more recent chat messages.

Users' Assessments of the System's Functionality. Some chat messages referred to users commenting on the app's functionalities and the operator's behavior. For example, there were several instances during the experiment when users complained that the commands issued by the operator were not representative of the chat messages. These types of complaints suggested either that the operator issued arbitrary commands ("nobody said to make tea," or "make tea was a super old command," while the agent was making tea) or that the experiment was scripted ("this is all scripted," or "lol this is mega scripted"). Users reacted quickly and vehemently to cases in which the operator started issuing commands that seemed to be completely unrelated to the chat. They were far more patient and lenient when the operator picked up on some real signal—no matter how small—in the chat log.

User Behavioral Patterns. There were several notable user behavioral patterns worth mentioning.

- *The obscene*: The user asks the agent to perform something the user believes funny, often in violation of public decency, such as "undress."
- *The saboteur*: The user asks the agent to perform something unrelated to the events taking place (for example, "say 'yeet'"). This was usually associated with an attempt to force the crowd conversation to ensure the command reached the agent (for example, reissuing the commands several times, using capitalization, and so on).
- *The joker*: The user asks the agent to perform something relatively unusual ("shout 'viva la Peru!'") but does not do it insistently
- *The emotional*: Users had various strategies for expressing emotions— using a foreign language, capitalization, and duplicating letters (for example, "screammmmm," "muuuuuug.")

Overall, it is essential to better understand these qualitative patterns because any automated text-to-command system that does not handle them properly runs the risk of missing important social, cultural, or contextual information. This issue is likely to be ameliorated by the use of machine learning and larger training sets.

Matching Functions

We present simple algorithms that can emulate the operator's behavior as a first step to designing smoother language-based systems for distributed control of human or machine behavior. Commands issued by the operator and suggested by the users were preprocessed (filtering, lemmatization, and part-of-speech tagging), typically producing a bigram consisting of a verb and a noun. We used the preprocessed commands issued by the human operator ($\mathbf{Y}$) to come up with simple but effective heuristics that accurately summarize online chats into a vector of

commands (**y**). Formally, the task is to maximize the proportion of matches between the human-generated commands (after preprocessing) **Y** and the machine-generated bigrams *f*(**X**) = **y**:

$$\frac{1}{N}\left[\sum_{i=1}^{N} f_k(X_i) = Y_i\right], \qquad (1)$$

where X_i is the chat data relative to a segment *i*, and $f_k(X_i)$ is an algorithm k operating on the segment's chat input data. Given the small quantity of data available, the findings presented here must be interpreted as a proof of concept rather than an exhaustive investigation. We experimented with various heuristic rules to get a coarse understanding of the data rather than trying to overfit any of the following methods. To avoid overfitting, we split the data into training and test sets. As the game produced two similar data streams (two agents and two crowds), we selected, at random, one agent (the character named *Winter*) to be used as the training set and left the other (the character named *Neuro*) as the test set.

We describe simple but useful heuristics that turned out to work surprisingly well (see the supplementary material for details). Code for reproducing the analysis is available on GitHub. We experimented with six different matching functions of increasing complexity [Figure 2(b)]:

- *Full match*: A naive, simple match. The most frequent bigram is compared against the operator's action command for a full string match. The algorithm does not deal very well with paraphrased commands.
- *Loose match*: The most frequent *q* bigrams are compared against the action command for a full string match. It is, essentially, a full match extended to include the *q* most popular bigrams (here, *q* = 5).
- *Bag-of-words match*: This function considers both the action and chat commands to be an unordered bag of words. The match score is the fraction of words in the preprocessed action command that is found in the candidate.
- *Indicator match*: A discretized form of the bag-of-words match. It returns one if at least half of the command words are found in the *q* most popular bigrams.
- *Moment match*: An indicator match on the first *q* bigrams, with a recency weight. Bigrams are sorted by their moment, which is calculated as $M(b) = \sqrt{\sum_i t_i}$, where t_i denotes the time of the ith occurrence of the bigram *b* in the segment.
- *Decay match*: An indicator test with a different ranking rule. A weighted sum is computed for each bigram, where weights are proportional to a square-rooted harmonic function: $\alpha \Big/ \sqrt{(\boldsymbol{d} + \beta)}$, where β controls a recency bias.

These functions can be seen as increasing in complexity. We present the match scores both when considering all segments and after removing segments when the operator issued an arbitrary command, which is not within the scope of this article. A segment was considered arbitrary (and classified as singleton) if the command issued by the operator was present only once or not at all in the chat. As we are interested in designing democratic algorithms, these are cases that we are not interested in modeling (and were actively criticized by our users, as described previously).

The results (Figure 3) show that the full match achieved the lowest performance (about 5.5% and 21% on test set, before and after singleton removal) due to its lack of flexibility. However, it performed better than a random model. We observe a sharp increase in performance when we relax the constraints for a match. This heuristic rests on the assumption that the meaning of a verb as well as the range of actions to perform with an object tend to be limited within a single context (for example, "turn left" versus "turn" are both considered matches for the command "turn left"). This function seems to reduce differences among similar (but not identical) users' suggestions and was shown to have little downside. Relaxing the definition of a full

Originally published in: Computer, 53(8), 2020, p. 54

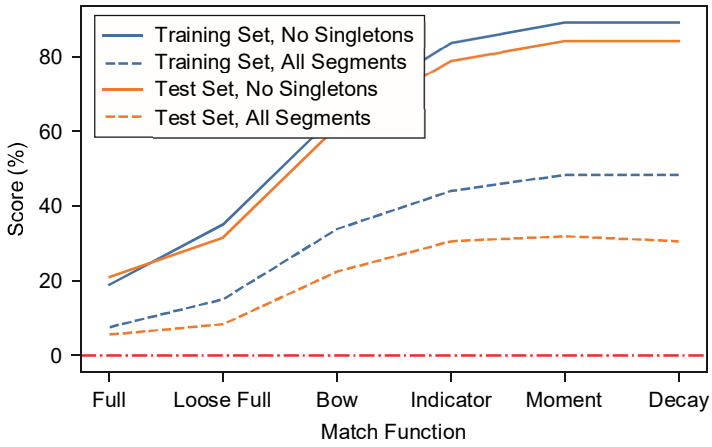

FIGURE 3. The performance (matching frequency score) of each match function tested as a function of training and test sets (color) and before or after singleton removal (line style). We report performance on both training and test sets because these correspond to two distinct crowds during the live event, each controlling a different character. Each set was selected at random. Notice that, although the match functions increase in complexity from left to right, the drop in performance between training and test sets is quite small (especially after removing singletons), suggesting little overfitting. The dot–dashed line represents chance performance (~2%).

match led to an increase in accuracy, both on raw data and after removing singletons (loose match: 8% and 31%; bow: 22% and 61%, respectively). Algorithms with a recency weight (moment and decay) performed best (moment: 31% and 84%; decay: 30% and 84%, respectively). Matching scores are all expressed as a proportion of all segments and can, thus, be compared with each other. Notice that training and test scores are similar after singletons are removed, suggesting little overfitting.

Real-Time Aggregation. The previous analysis used labeled data—where labels are the command issued by the operator—to define a set of simple algorithms that can emulate human performance. It relies on the segmentation of a continuous chat stream into discrete units (segments), which, by definition, can only happen offline. However, if we want to design an algorithm to replace the human operator, we cannot rely on labeled data, as segments are undefined. Instead, the algorithm should be able to read chat logs in real time and dynamically aggregate them into a relevant command. During the BeeMe event, the operator could see when the agent completed a requested action from the video stream and could then issue a new command. Similarly, such an algorithm should output the most wanted command whenever an action slot becomes available (for example, when the agent completes the previous action) or a popularity threshold is reached. For this purpose, we implemented a simple online chat aggregator. The algorithm dynamically updates a tally of all available bigrams (derived from users) and ranks them by relevance.

The tally score S for each bigram b is computed on the basis of the decay match test as

$$S_b = \sum_{i=1}^{M} \frac{\alpha}{\sqrt{(d_i + \beta)}}, \qquad (2)$$

where α is a constant (here, set to 1,000), β is a decay dampener controlling the rate of decay for the relevance of each bigram's occurrence i (set to 5,000), d is the time difference between the current time and each occurrence time stamp ($t_0 - t_i$), and M is the total number of occurrences of a bigram, namely, all messages mapping onto the same bigram b.

The results (Figure 4) show the score time series of the top three bigrams at each time step. The colors represent different bigrams. For reference, we have annotated the original chat messages corresponding to noticeable scoring peaks. It can be seen that score peaks represent highly relevant commands corresponding to popular user suggestions. Many of these (for example, "go to building 7") were commands issued by the operators during the Halloween event.

DISCUSSION

In this article, we presented a new online platform, named *BeeMe*, designed to experiment with distributed control of human action in open-ended decision environments. We used data from a global performance that went live in 2018 as a first case study. We showed how simple but effective algorithms can represent a crowd's intentions by democratically aggregating linguistic action suggestions in real time.

Our first finding was that many commands issued by the operator to the agent were not represented in the chat data that preceded them. One

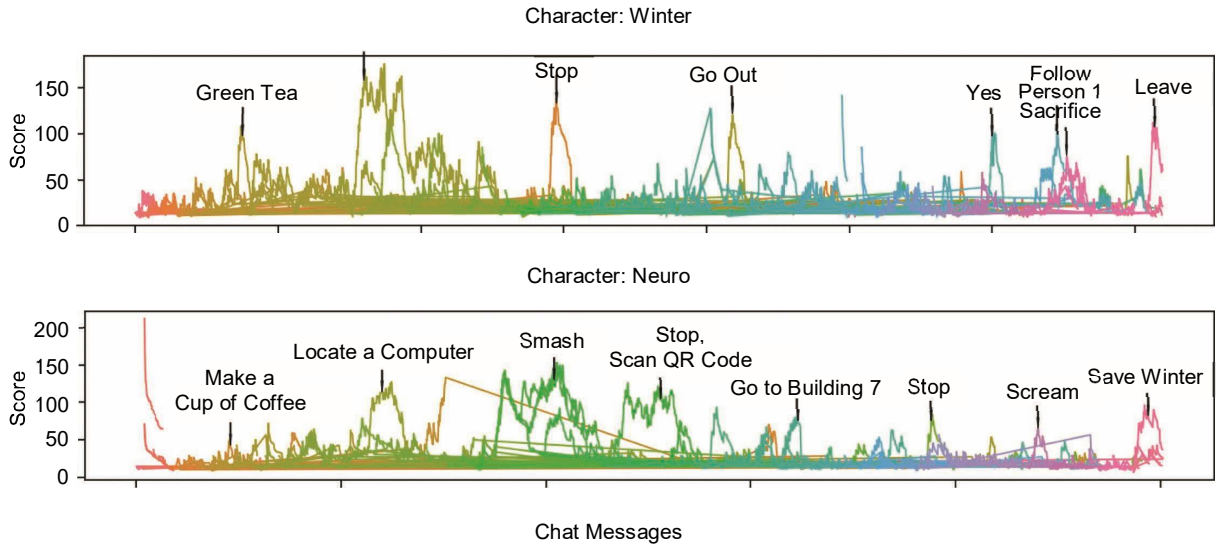

FIGURE 4. The online aggregator tally score for each bigram (*y*-axis) over chat messages exchanged by the crowd (*x*-axis). The algorithm updates a dynamic tally of all bigrams *b* produced by the crowd and scores them as a weighted sum over all bigram instances *i*. The weight associated with each instance is related to the time difference from the current time and the time stamp associated with the original instance.

explanation is that operators were simply not paying attention due to cognitive overload. Alternatively, operators might have been actively trying to sway the conversation in their preferred direction (for example, making the story continue, helping a stuck agent, and so on).

Second, algorithms that loosely match human-generated commands outperform algorithms that try to exactly match human operators. Although the algorithm might have missed some important contextual or linguistic information, the fault might also be on the human side. Either the operators were not able to keep a perfect count of each command suggestion, or they might have been consciously trying to ignore some crowd's suggestions. Notice that, in Figure 3, the ground truth is the operator's behavior. This introduces a fundamental problem when trying to create more democratic algorithms that improve on human biases. Introducing objective performance metrics (such as user satisfaction, efficiency in completing the narrative, and so on) will be crucial in future data collection to design truly democratic algorithmic aggregators.

Collective control through automatic opinion aggregation is an intriguing avenue of investigation in hybrid intelligence that might have far-reaching consequences in many domains, from gaming and navigation to democratic representation.[7] Scalable hybrid conversational decision systems leverage the flexibility and transparency of human language, but challenges remain, like scaling up argumentation and extending decisions to complex problem solving.[9,16] Similarly, the study of algorithmic consensus facilitation is highly timely, given the increasing pace of collective attention.[17] Algorithms that facilitate the functioning of institutions and the enforcement of social norms— from fake news and hate speech algorithmic moderation to Wikipedia bots fighting vandalism—feature low cost, scalability, and speed. Although many pitfalls have also been documented, including biases and lack of accountability,[18] integrating human and machine intelligence promises to overcome many limitations of nonhybrid systems.

The ability to quickly formulate the best course of action in the presence of dynamic environments and shifting goals lies at the heart of intelligent behavior. Both biological intelligence and AI can achieve remarkable results in strategizing and planning. Although collectives exhibit remarkable intelligence,[11–13] whether they can also show complex problem-solving abilities is still poorly understood.

Originally published in: Computer, 53(8), 2020, p. 56

Our work adds to a list of notable Internet performances, including Reddit Place and Twitch Plays Pokémon. These unique performances are of interest for practitioners in collective decision making and human–computer interaction because they highlight how goal-oriented behavior, typical of higher cognitive functions, can emerge in a decentralized manner from distributed social systems.[19,20] However, BeeMe is designed around natural language to achieve real-time democratic consensus instead of the simpler aggregation typically seen in previous Internet performances. It is, thus, better suited for open-ended task or action sets and for problems requiring deliberation without the need for argumentation mapping.[16]

Finally, we draw attention to the importance of such massive gamified social experiments. Although difficult to replicate, these events are valuable per se because they embody a proof of existence, highlighting the potential (and limitations) of collective intelligence in the real world. They spark the imaginations of users as well as practitioners interested in improving collective intelligence design.

Although the results presented here are in no way final and, thus, to be interpreted with caution, collection of more data through future BeeMe performances will help in understanding how hybrid design can facilitate realtime, goal-oriented coordination of large groups. To this end, we encourage the participation of academics and artists to suggest their own performances using the BeeMe platform. Submissions should be addressed to Niccolo Pescetelli. They will be evaluated, and the most promising will be implemented in collaboration with the creators.

ACKNOWLEDGMENTS

The authors would like to thank Samantha Miller, who contributed to the development of the web environment and to the design of the Halloween event; Arthur Pemberton, who developed the BeeMe platform; and Peter E. Hussami, who helped develop the algorithms reported in this article. The authors would also like to thank the Massachusetts Institute of Technology Media Lab and Institute for Data, Systems, and Society (IDSS), which funded and hosted this project.

Reproducibility

Data and code to reproduce the analysis are available via GitHub at https://github.com/chri4354/beeme. The platform code is publicly available at https://github.com/chri4354/ BeeMe_platform.

Contributions

Most activities for this article, including the concept design, design of the platform, data collection, and analysis, were done when the authors were at the Massachusetts Institute of Technology, Media Lab.

N. Pescetelli and M. Cebrian designed and conceptualized the research and directed the live demonstration. I. Rahwan supervised the live demonstration. All the authors developed formal methods and analyzed the data. Pescetelli wrote the manuscript, and Cebrian and Rahwan edited the manuscript.

REFERENCES

1. M. Tomasello, *Why We Cooperate*. Cambridge, MA: MIT Press, 2009.
2. P. Richerson and J. Henrich, “Tribal social instincts and the cultural evolution of institutions to solve collective action problems,” *SSRN Electron. J.*, vol. 3, no. 1, pp. 1–29, 2009. doi: 10.2139/ssrn.1368756. [Online]. Available: http://www.ssrn.com/abstract=1368756
3. L. B. Rosenberg, “Human Swarms, a real-time method for collective intelligence,” in *Proc. Eur. Conf. Artificial Life 2015*. Cambridge, MA: MIT Press, 2015, pp. 658–659. doi: 10.7551/978-0-262-33027-5-ch117. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/978-0-262-33027-5-ch117
4. M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger, “Crowds in two seconds: Enabling realtime crowd-powered interfaces,” in *Proc. 24th Annu. ACM Symp. User Interface Software and Technology (UIST’11)*, 2011, pp. 33–42. doi: 10.1145/2047196.2047201.
5. V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. doi: 10.1038/nature14236.
6. A. Dafoe, “AI governance: A research agenda,” in *Governance of AI Program, Future of Humanity Institute*. Oxford, U.K.: Univ. of Oxford, 2018.
7. S. Gouravajhala, J. Yim, K. Desingh, Y. Huang, O. Jenkins, and W. Lasecki, “EURECA: Enhanced understanding of real environments via crowd assistance,” in *Proc. AAAI Conf. Human Computation (HCOMP)*, Zurich, Switzerland, 2018, pp. 31–40.
8. L. Chen, D. J. Cook, B. Guo, L. Chen, and W. Leister, “Guest editorial special issue on situation, activity, and goal awareness in cyber-physical human–machine systems,” *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 3, pp. 305–309, 2017. doi: 10.1109/THMS.2017.2689178. [Online].

Available: http://ieeexplore.ieee.org/document/7927788/
9. D. Dellermann, P. Ebel, M. Söllner, and J. M. Leimeister, “Hybrid intelligence,” *Bus. Inform. Syst. Eng.*, vol. 61, no. 5, pp. 637–643, 2019. doi: 10.1007/s12599-019-00595-2.
10. G. Demartini, D. E. Difallah, U. Gadiraju, and M. Catasta, “An introduction to hybrid human-machine information systems,” *Found. Trends Web Sci.*, vol. 7, no. 1, pp. 1–87, 2017. doi: 10.1561/1800000025. [Online]. Available: http://www.nowpublishers.com/article/Details/WEB-025
11. I. D. Couzin, “Collective cognition in animal groups,” *Trends Cogn. Sci.*, vol. 13, no. 1, pp. 36–43, 2009. doi: 10.1016/j.tics.2008.10.002. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19058992
12. A. Dorri, S. S. Kanhere, and R. Jurdak, “Multi-agent systems: A survey,” *IEEE Access*, vol. 6, pp. 28,573– 28,593, Apr. 2018. doi: 10.1109/ ACCESS.2018.2831228. [Online]. Available: https://ieeexplore.ieee.org/document/8352646/
13. L. Rosenberg, D. Baltaxe, and N. Pescetelli, “Crowds vs swarms, a comparison of intelligence,” in *Proc. 2016 Swarm/Human Blended Intelligence (SHBI 2016)*, pp. 1–4.
14. “MIT invites you to control a human on Halloween,” *BBC Technology*, 2018. [Online]. Available: https://www.bbc.com/news/business-46016696
15. P. Harrigan and N. Wardrip-Fruin, *Second Person: Role-Playing and Story in Games and Playable Media*. Cambridge, MA: MIT press, 2010.
16. L. Iandoli, M. Klein, and G. Zollo, “Enabling on-line deliberation and collective decision-making through large-scale argumentation: A new approach to the design of an internet-based mass collaboration platform,” *Int. J. Decis. Support Syst. Technol.*, vol. 1, no. 1, pp. 69–92, 2009. doi: 10.4018/jdsst.2009010105. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve .aspx?doi=10.4018/978-1-60566-727-0 http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdsst.2009010105
17. P. Lorenz-Spreen, B. M. Mønsted, P. Hövel, and S. Lehmann, “Accelerating dynamics of collective attention,” *Nat. Commun.*, vol. 10, no. 1, p. 1759, 2019. doi: 10.1038/s41467-019-09311-w. [Online]. Available: http://www.nature.com/articles/s41467-019-09311-w
18. N. Diakopoulos and S. Friedler, “How to hold algorithms accountable,” *MIT Technology Rev.*, Nov. 17, 2016. [Online]. Available: https://www.technologyreview.com/2016/11/17/155957/how-to-hold-algorithms-accountable/
19. A. Aleta and Y. Moreno, “The dynamics of collective social behavior in a crowd controlled game,” *EPJ Data Sci.*, vol. 8, no. 1, p. 22, 2019. doi: 10.1140/epjds/s13688-019-0200-1.
20. T. F. Müller and J. Winters, “Compression in cultural evolution: Homogeneity and structure in the emergence and evolution of a large-scale online collaborative art project,” *PLoS One*, vol. 13, no. 9, p. e0202019, 2018. doi: 10.1371/journal.pone.0202019. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0202019
21. Center for Humans and Machines, “The BeeMe experiment,” Vimeo. [Online]. Available: https://vimeo.com/370031909

ABOUT THE AUTHORS

NICCOLO PESCETELLI is a principal investigator at the Center for Humans and Machines at the Max Planck Institute for Human Development, Berlin, Germany. His research interests include collective decision making. Pescetelli received a D.Phil. in experimental psychology from the University of Oxford, United Kingdom. Contact him at pescetelli@mpib-berlin.mpg.de

MANUEL CEBRIAN is a research group leader at the Center for Humans and Machines at the Max Planck Institute for Human Development, Berlin, Germany. His research interests include time-critical social mobilization. Cebrian received a Ph.D. in computer science from the Universidad Autonoma de Madrid, Spain. Contact him at cebrian@mpib-berlin.mpg.de.

IYAD RAHWAN is the founder and director of the Center for Humans and Machines at the Max Planck Institute for Human Development, Berlin, Germany. Rahwan received a Ph.D. in information systems from the University of Melbourne, Australia. Contact him at rahwan@mpib-berlin.mpg.de.

Originally published in: Computer, 53(8), 2020, p. 58