Real-time Internet Control of Situated Human Agents

NICCOLO PESCETELLI, Center for Humans & Machines, Max-Planck Institute for Human Development, Berlin, Germany

MANUEL CEBRIAN, Center for Humans & Machines, Max-Planck Institute for Human Development, Berlin, Germany

IYAD RAHWAN, Center for Humans & Machines, Max-Planck Institute for Human Development, Berlin, Germany

We present an online platform, called BeeMe, designed to test the current boundaries of Internet collective action and problem solving. BeeMe allows a scalable internet crowd of online users to collectively control the actions of a human surrogate acting in physical space. BeeMe demonstrates how intelligent goal-oriented decision-making can emerge from large crowds in quasi real-time. We analyzed data collected from a global BeeMe live performance that involved thousands of individuals, collectively solving a sci-fi Internet mystery. We study simple heuristic algorithms that read in users' chat messages and output human actionable commands representing majority preferences, and compare their performance to the behavior of a human operator solving the same task. Results show that simple heuristics can achieve near-human performance in interpreting the democratic consensus. When human and machine's output differ, the discrepancy is often due to human bias favoring non-representative views. We discuss our results in light of previous work and the contemporary debate on democratic digital systems.

CCS Concepts: • **Human-centered computing** → *Open source software.*

Additional Key Words and Phrases: collective intelligence, collective action, Internet control, hybrid intelligence, heuristics

ACM Reference Format:
Niccolo Pescetelli, Manuel Cebrian, and Iyad Rahwan. 2018. Real-time Internet Control of Situated Human Agents. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Humans are extreme cooperators [18, 20]. The evolution of language allowed our species to coordinate actions, thoughts and intentions. Contrary to other social species — like eusocial insects and other primates — humans cooperate in large-scale societies of largely non-related individuals [4, 6, 8]. The increasing returns of group scale, evident for instance in warfare, big game hunting, or tax collection, has put pressure on humans to develop norms and institutions that allow increasingly larger social groups to cooperate and act together [3, 21, 22]. Social norms and institutions— defined as the set of culturally acquired rules governing the behavior of individuals in specific contexts [21]— are considered solutions to the collective action problem, arising when selfish instincts must be suppressed in favor of higher group payoffs [8, 13, 17]. Religion and political states are thought to have provided a range of innovations— like moralizing gods, policing and representative hierarchies — that maintain fairness, punish non-cooperators and increase overall society's stability [2, 12, 16]. Importantly, the scale at which such insititutions and cultural norms change in response to new goals or environments is often in the range of years to generations. Digital technologies of the Internet era promise to greatly speed up the time scale at which large collectives can demonstrate intelligent behavior [19]. But the main challenge in creating real-time collective decision systems remains how to interpret the intentions of a crowd.

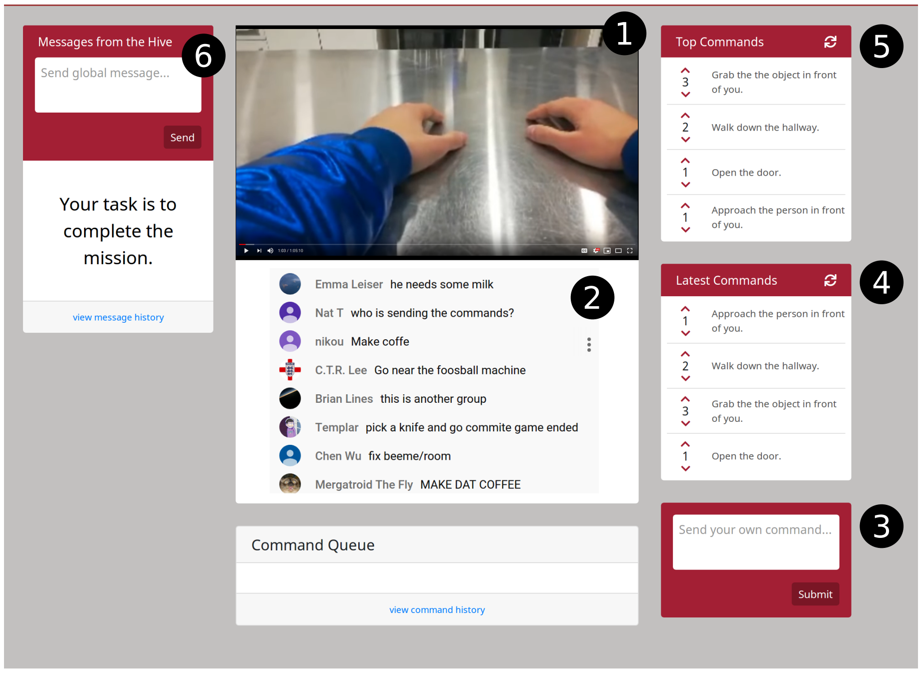
Artificial intelligence has made great progresses in the last decade, thanks to the advancement of computing, data access and economy of scale. The use of algorithms to replicate human functions has been extended to include several new domains from perception to art, language and situated action [10, 14]. Whether machines can be used for the betterment of our political institutions and or the enforcement of social norms remains unclear.

In this paper, we describe a platform that we created as a playful attempt to experiment with new creative solutions to scalable collective action. The platform, called BeeMe, is accessible at https://beeme.online. BeeMe allows mutliple users to observe and collectively control the actions of one human surrogate (the agent) operating in physical space during scheduled events. We analyze data obtained during one of such events, which went live on Halloween night 2018, when thousands users played simultaneously. Unbeknown to them, users were randomized in two different crowds, each controlling one agent. Online users had to coordinate their suggestions to the agent using an in-built chat and/or a rank voting system, in order to solve a 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 [7, 15]. Another experimenter's confederate (the operator), aggregated users' suggestions into single commands and relayed them to the agent. The operator's role was thus to interpret the crowd's 'will' 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. In this paper we want to understand (1) how a human interpreter aggregated online discussions in real time and (2) whether an algorithm can be implemented to achieve the same or better performance than a human.

Results suggest that human operators were often biased and did not follow the majority when interpreting the crowd's will. For example, they tended to relay commands that were only mentioned once and thus not representative of the majority opinion (here called 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 in human discussions (chat logs) and map them into relevant action commands representing the democratic view in the crowd. We find that machine-generated commands often loosely match human-generated commands, *i.e.*, human-generated commands shows among the first five commands ranked by the algorithm. When the human and machine generated commands differ, machine-generated commands are still sensible (*i.e.*, actionable) given the recent past chat discussion. We discuss these findings in light of the recent events surrounding politics and online algorithms.

2 THE PLATFORM

The BeeMe platform was designed to allow a large internet crowd to remotely control the actions in physical space of one agent, in order to complete simple games (here a sci-fi mystery). The online user interface (Figure 1) 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 (like "turn left" or "run away"); (4-5) a voting mechanism that users can use to up-vote and down-vote others' commands, based on recency (4) or popularity (5); (6) an experimenter text box, where the experimenter could send messages to all users to give the crowd new missions or advance the game's narrative. Winning commands were transmitted to the agent(s) by an operator who was monitoring the web-app and was in contact with the agent via an earpiece. Unbeknownst to them, online users were randomly assigned to two teams each controlling a different character in the story (*i.e.*, a different agent) who was freely roaming the MIT campus. The game consisted of a series of 11 missions that had to be carried out under time pressure to defeat the characters' foe. 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 had thus 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 whether an algorithm reading in a stream of chat messages could achieve human-level results.



Emma Leiser he needs some milk
Nat T who is sending the commands?
nikou Make coffe
C.T.R. Lee Go near the foosball machine
Brian Lines this is another group
Templar pick a knife and go commite game ended
Chen Wu fix beeme/room
Mergatroid The Fly MAKE DAT COFFEE

Top Commands:
3 Grab the the object in front of you.
2 Walk down the hallway.
1 Open the door.
1 Approach the person in front of you.

Latest Commands:
1 Approach the person in front of you.
2 Walk down the hallway.
3 Grab the the object in front of you.
1 Open the door.

Fig. 1. Schematic of the BeeMe platform interface. **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 upvote or downvote each others' commands based on latest suggestions; **5**) a running tally of most popular commands is also showed to users for control actions that are about to be communicated to the agent; **6**) An experimenter box allows admin users to send messages to online users to communicate new missions and advance the narrative. *Picture obtained from the Halloween 2018 BeeMe event, (c) N. Pescetelli.*

3 THE DATA

For the purpose of analysis, there were two data streams:

(1) The chat log – 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.

(2) The actions/commands issued by the human operators and performed by the agent during the live event, represented by a text body and a time stamp.

We segmented the chat data based on the commands issued by the operators (Figure 5). The term action "segment" will denote a period of time that encompasses the time interval between two consecutive actions. The event lasted for about one hour. The total number of actions performed by the two agents was 91 and 84 for the two characters respectively. The small number of data points did not allow the use of sophisticated NLP techniques (*e.g.*, using deep learning). Instead, we opted for a more viable rule-based approach, consisting in reducing all commands and chat messages to *(noun + verb)* pairs and performing statistical tests on those. We fitted our six algorithms to one character's data, selected at random, and used the second character's data as test set.

4 QUALITATIVE OBSERVATIONS

Both crowds managed to coordinate their suggestions and to complete all 11 missions that the game entailed. This suggests that distributed collective control was possible notwithstanding the adversarial effects of communication delays, trolling and social miscoordination. Second, visible patterns emerged during the experiment. These are worth mention as they can help formulate the model and/or suggest questions for further study. The ones discussed here are:

- Frequent chat commands were often the winners
- Command spillover
- Users' assessment of the system's functionality
- User behavioral patterns

4.1 Frequent suggestions were often the winners

Visual inspection showed that for a large cluster of actions (36% and 22% respectively), the command that the agent received was one of the most frequent commands suggested by the crowd. The fact that a popular, recent command is the one that ends up executed is the expected behavior from the system that we are trying to model.

4.2 Spillover effects

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

A second form of spillover was operator's spillover. This happens when, between two similarly popular commands on segment k (*e.g.*, "make tea" and "leave the room"), the operator chooses one as the action to perform (*e.g.*, "make tea"). Then, once that action is performed, they transmit the second one (*e.g.*, "leave the room"), on segment $k + 1$, disregarding more recent chat messages.

4.3 Users' assessment of the system's functionality

Some chat messages referred to users commenting on the app's functionalities and the operator's behavior. For instance, 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 started making tea) or that the experiment was scripted ("this is all scripted", "lol this is mega scripted" etc.). Users reacted quickly and vehemently to cases where 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.

4.4 User behavioral patterns

There were several notable user behavioral patterns worth mentioning.

- *The obscene.* The user asks the agent to perform something "funny", often in violation of public decency. *e.g.*"undress".
- *The saboteur.* The user asks the agent to perform something unrelated to events taking place (*e.g.*"say 'yeet'"). This was usually associated with the attempt to force the crowd conversation to ensure the command reached the agent (*e.g.*, re-issuing the commands several times, using capitalization, etc.).
- *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, duplicating letters. *e.g.*"screammmmm", "muuuuuug" etc.

It is essential to better understand these 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 current machine-learning tools and larger training sets.

5 MATCHING FUNCTIONS

The goal of this work was to design simple algorithms that can emulate the operator's behavior. This is a first step to design smoother language based systems for distributed control of human or machine behavior. Commands issued by the operator and suggested by the users were lemmatized, typically producing a bigram consisting of a verb and a noun. We used the (pre-processed) commands issued by the human operator (Y) to come up with simple but effective heuristics that accurately summarize online chats into a series of commands (y). Formally, the task is to maximize the proportion of matches between the human generated commands (after pre-processing) Y and the machine generated bigrams $f(X) = y$:

$$\frac{1}{N}(\sum_{i=1}^{N} f_k(X_i) = Y_i) \tag{1}$$

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 below must be interpreted as a proof of concept rather than an exhaustive investigation on the matter. Our approach consisted in experimenting 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 splitted the data in a training and test sets. As the game produced two similar data streams (two agents and two crowds), we selected at random one agent (character "Winter") to be used as the training set and left the other (character "Neuro") as a test set.

We describe below simple but useful heuristics that turned out to work surprisingly well. Code for reproducing the analysis is available on Github. We experimented with six different matching functions of increasing complexity (Figure 2):

- *Full match.* The full match is a very naive, simple match: the most frequent bi-gram is compared against the operator's action command for a full string match, return 1 if the match is true, otherwise 0. The algorithm does

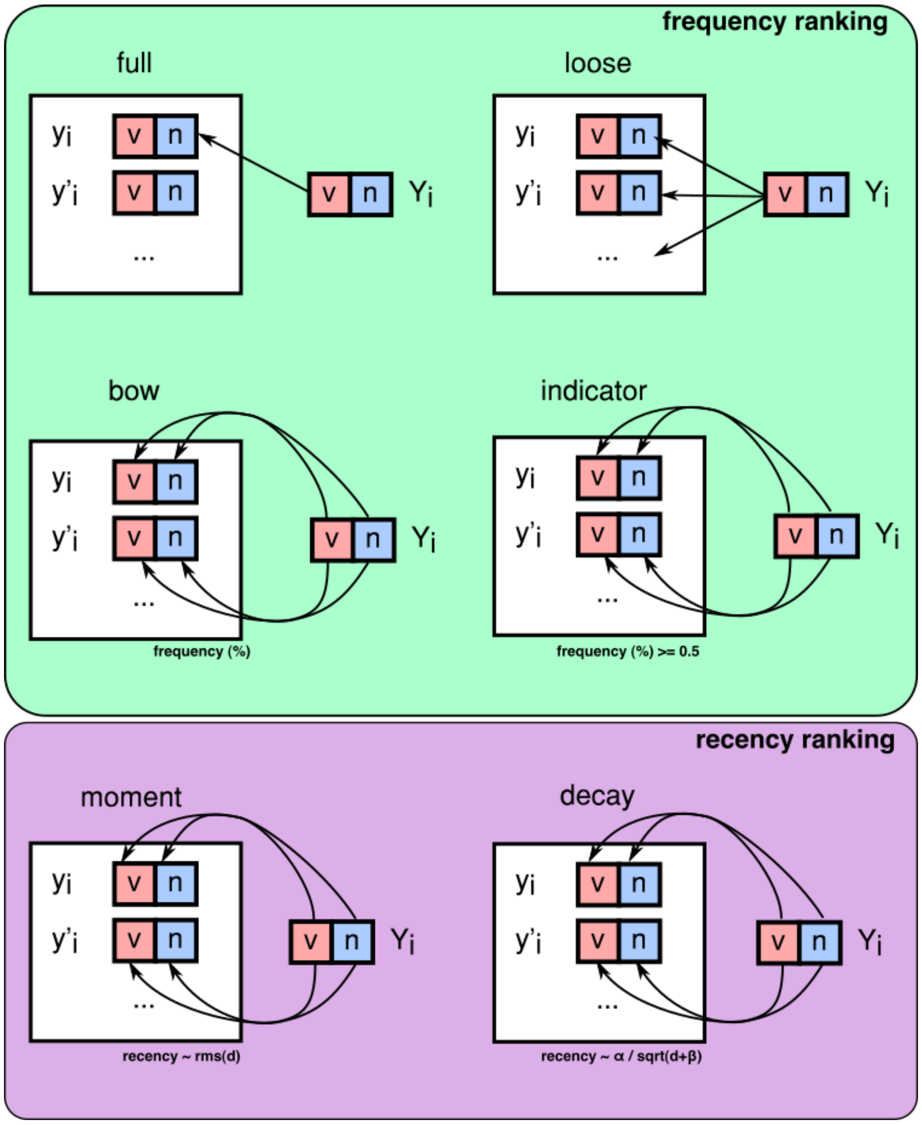


Fig. 2. Graphical representation of the six algorithms tested. Each algorithm ranks the set of available bigrams generated from the chat, based on frequency (green panel) and recency (purple panel). Arrows, represent comparisons/matches between operator issued bigrams **Y** and bigrams obtained from users' suggestions (**y**). Comparisons were either between entire bigrams (first row) or words within bigrams (second and third row).

not deal very well with paraphrased commands or with commands that have a word count different from 2 (after pre-processing).

- *Loose match.* The most frequent k bi-grams are compared against the action command for a full string match, returns 1 if the match is true otherwise 0. It is essentially a full match extended to include k most popular bi-gram (here $k = 5$).

- *Bag of words match.* This function considers both the action command and the chat command to be an unordered "bag" of words. The match score for a bi-gram is the fraction of words in the pre-processed action command that are found in the candidate with a full string match. It returns the highest score over the most popular k bi-grams.
- *Indicator match.* A discretized form of the bag of words match: returns 1 if and only if at least half of the action command's words are found in a bi-gram, over the k most popular bi-grams.
- *Moment match.* An indicator match on the first k bi-grams, sorted by their "moment", which is calculated as $M(b) = \sqrt{sum_i t_i}$, where t_i denotes the time of the i-th occurrence of the bi-gram b in the segment.
- *Decay match.* An indicator test with a different ranking rule. After removing messages above a certain time threshold (here 50 seconds), a weighted sum is computed for each bi-gram, where weights are proportional to a square rooted harmonic function: $\alpha / sqrt(d + \beta)$. Here, α is a constant ($\alpha = 1000$), d is the time difference between the bi-gram occurrence i and the issued action, and β is a dampening constant that controls the bias towards instances that are very close to end of the segment (here, $\beta = 5000$).

The functions above can be seen as increasing in complexity. The *indicator match*, which produces far better results than the functions preceding it in the above ordering, might seem counter-intuitive at first. The rationale for matching at least half the words of the action command rests on the hypothesis that a 50%+ match of the action command's words means that either the action verb or the object noun is matched correctly. 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 (*e.g.*, "turn left" vs. "turn" are considered both matches for the command "turn left"). In practice, we observed this to work very well – this function seemed to reduce differences among similar (but not identical) users' suggestions and showed to have little downsides.

The *decay match* is similar to the *moment match*, which sorts the segment's bi-grams and performs a modified indicator test on the top candidates. The *decay* test however is a more sophisticated form of the *moment* test that does not rely on segmentation, and can thus support online computation mode (see *Online aggregation*, below). Results are described below.

6 RESULTS

6.1 Match results

We present the results of the match functions both when considering all segments and after removing segments when the operator issued an arbitrary command, which are not of interest for the scope of this paper. A segment was considered arbitrary (and classified as "singleton") if the command issued by the operator was present only once or less than once in the chat. These are limit cases that we are not interested in modeling. Results, (Figure 3) show that the $full$ match achieved the lowest performance (about 5.5% and 21% on test set, before and after removing singleton segments), due to its lack of flexibility. Relaxing the definition of a full match, led to an increase in accuracy both on raw data and after removing singletons ($loosematch$: 8% and 31% respectively, bow: 22% and 61%). Algorithms with a recency weight ($moment$ and $decay$) perform best ($moment$: 31% and 84%; $decay$: 30% and 84% respectively). Matching score are all expressed as a proportion and can thus be compared with each other. Notice that training and test scores are similar, after removing singletons, suggesting that overfitting was likely not an issue.

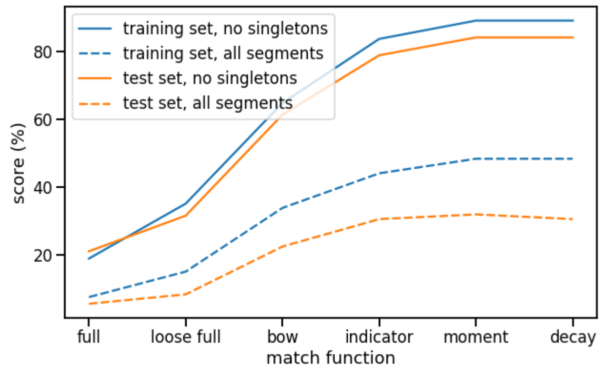


Fig. 3. Performance (matching frequency score) of each match function tested, as a function of training and test sets (color), and before or after singleton removal (line style). We report performance on both training and test sets because these corresponds to two distinct crowds during the live event, 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 no overfitting.

6.2 Online 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 (*i.e.*, proportion of matching segments). 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 in real-time chat logs 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 algorithm should output the most wanted command, whenever an action slot becomes available (*e.g.*, when the agent completes the previous required action) or whenever a popularity threshold is reached. For this purpose, we implemented a simple online chat aggregator. The algorithm dynamically updates a tally of all available bi-grams (derived from users' suggested commands) and rank them by relevance.

The tally score S for each bi-gram b is computed on the basis of the *decay match* test as:

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

where α is a constant (here set to 1000.0), β is a decay dampener controlling the rate of decay for the relevance of each bi-gram's occurrence i (set to 5000), d is the time difference between the current time and each occurrence timestamp ($t_0 - t_i$) and M is the total number of occurrences of a bi-gram, namely all messages mapping onto the same bi-gram b.

Results (Figure 4) show the score timeseries of the top three bi-grams at each timestep. Color represents different bi-grams. For reference, we have annotated original chat messages corresponding to noticeable scoring peak. It can be seen that score peaks represent highly relevant commands corresponding with popular users' suggestions. Many of these (*e.g.*, "go to building 7") were commands issed by the operators during the Halloween event.

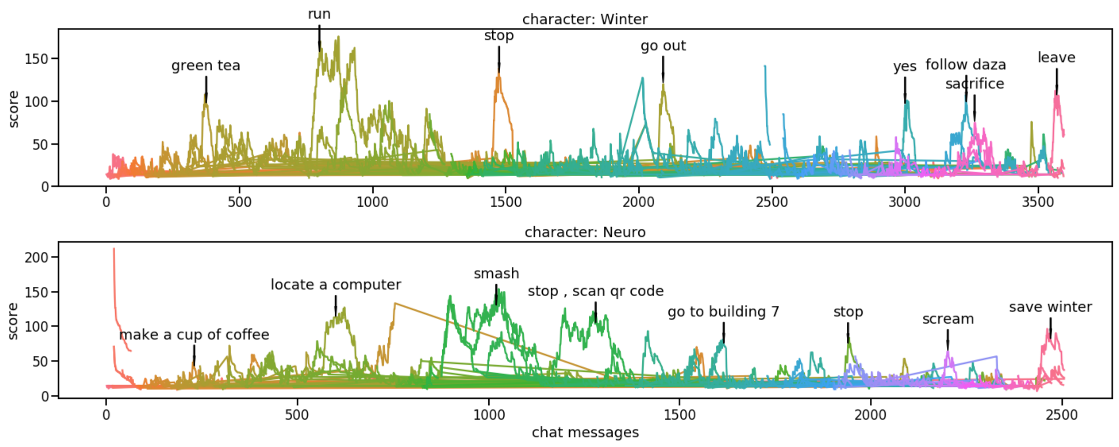


Fig. 4. Online aggregator tally score for each bigram (y-axis) over chat messages exchanged by the crowd (x-axis). The algorithm updates a dynamic tally of all bi-grams b produced by the crowd and scores them as a weighted sum over all bigram instances i. The weight associated with each instance is related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

7 DISCUSSION

Human ability for cooperation is remarkable. Digital technologies are transforming the way and the speed at which humans can coordinate their behavior to produce goal-oriented intelligent behavior. In this paper we presented some preliminary data collected from the BeeMe platform, a new collective action platform launched in 2018 for online streaming performances. We use data from one such global live events to investigate whether simple but effective algorithms can represent a crowd's 'will' by aggregating in real time democratic suggestions to an extent that mimics what a human operators would do.

Our first finding was that a large portion of commands issued by the operator to the agent were not represented (or represented only once) in the chat data that preceded them. This result is compatible with different explanations. The simplest is that operators were simply not paying attention to the chat until an action slot became available (*e.g.*, the agent completed the previous action and a new one was required). Alternatively, operators might have been attentive but not objective in their task, or actively trying to sway the conversation in their preferred direction (*e.g.*, making the story continue, helping a stuck agent etc.). This might also suggest that the time that took for crowds during the Halloween event to complete all 11 tasks (~an hour) is likely an upper bound for distributed collective control in these types of games.

Second, algorithms that loosely match human generated commands outperform algorithms that try to exactly match human operators (*e.g.*, *loose match* vs. *full match*). This suggests that when the algorithm did not exactly match the human command, the latter was likely found among the algorithm's second or third best option. 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 that were deemed to be inappropriate or dangerous for the agent.

Collective control through automatic opinion aggregation is an intriguing avenue of investigation that might have large-scale consequences in many online settings, from gaming to democratic representation. The ability to quickly formulate the best course of action after a change in goals or environmental state lies at the heart of intelligent behavior. Both individual biological intelligence and artificial intelligence can achieve remarkable results in this respect, by showing flexible strategizing and planning. Whether groups can show such ability was, until recently, unknown. Our work add to a list of notable Internet performances, including Reddit Place and Twitch Plays Pokémon, which highlighted how intelligent goal-oriented behavior, typical of higher cognitive functions, can emerge in a decentralized manner from distributed social systems [1, 11]. At the same time, our work employs a real-time democratic consensus system based on language, instead of simpler local aggregation rules typically used in other Internet performances.

The pace of online discussions and collective attention has accelerated over the recent years [9]. The study of algorithmic consensus facilitation and distributed decision making and control is both timely and relevant for our society. Many algorithms are already used in online platform that facilitate the good functioning of institutions and the enforcement of social norms, from fake news and hate speech algorithmic moderation to Wikipedia bots fighting vandalism. Many benefits characterize the use of such algorithms, like low cost, scalability and speed. However, many pitfalls have also been documented, including biases and lack of accountability [5].

Finally, we draw attention to the importance of such large scale gamified social experiments. Although unique and difficult to replicate, these events are valuable *per se* because they embody a proof of *existence*, highlighting the potential (and boundaries) of collective intelligence in the digital era. They spark the imagination 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, the availability of more data through future BeeMe performances will be pivotal in understanding the extent to which algorithms can facilitate real-time goal-oriented coordination of large groups. To this end, we encourage the participation of academics and artists to suggests their own experiments, performances and events, using the BeeMe platform. Submissions should be addressed to the corresponding author. They will be evaluated by the authors and the most promising will be implemented with the collaboration of the creators.

CONCLUSIONS

Cooperation among large collectives is considered one of the most important evolutionary achievements of our species. Digital technologies promise to change the time scale at which goal-oriented collective action can take place. Our preliminary results pave the way for future investigations of hybrid collective action systems.

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, and Peter E. Hussami, who helped developing the algorithms reported here.

REFERENCES

[1] Alberto Aleta and Yamir Moreno. 2019. The dynamics of collective social behavior in a crowd controlled game. *EPJ Data Science* 8, 1 (12 2019), 22. https://doi.org/10.1140/epjds/s13688-019-0200-1

[2] Carlos A. Botero, Beth Gardner, Kathryn R. Kirby, Joseph Bulbulia, Michael C. Gavin, and Russell D. Gray. 2014. The ecology of religious beliefs. *Proceedings of the National Academy of Sciences* 111, 47 (11 2014), 16784–16789. https://doi.org/10.1073/pnas.1408701111

[3] S. Bowles. 2009. Did Warfare Among Ancestral Hunter-Gatherers Affect the Evolution of Human Social Behaviors? *Science* 324, 5932 (6 2009), 1293–1298. https://doi.org/10.1126/science.1168112

[4] L. G. Dean, R. L. Kendal, S. J. Schapiro, B. Thierry, and K. N. Laland. 2012. Identification of the Social and Cognitive Processes Underlying Human Cumulative Culture. *Science* 335, 6072 (3 2012), 1114–1118. https://doi.org/10.1126/science.1213969

[5] Nicholas Diakopoulos and Sorelle Friedler. 2016. How to Hold Algorithms Accountable. *MIT Technology Review* (11 2016).

[6] Jared M. Diamond. 1997. *Guns, Germs, and Steel: The Fates of Human Societies.* W. W. Norton & Company, New York.

[7] P. Harrigan and N. Wardrip-Fruin. 2010. *Second person: Role-playing and story in games and playable media.* The MIT press, Cambridge, MA.

[8] J. Henrich, Jean Ensminger, R. McElreath, Abigail Barr, Clark Barrett, Alexander Bolyanatz, Juan Camilo Cardenas, Michael Gurven, Edwins Gwako, N. Henrich, C. Lesorogol, F. Marlowe, D. Tracer, and J. Ziker. 2010. Markets, Religion, Community Size, and the Evolution of Fairness and Punishment. *Science* 327, 5972 (3 2010), 1480–1484. https://doi.org/10.1126/science.1182238

[9] Philipp Lorenz-Spreen, Bjarke Mørch Mønsted, Philipp Hövel, and Sune Lehmann. 2019. Accelerating dynamics of collective attention. *Nature Communications* 10, 1 (12 2019), 1759. https://doi.org/10.1038/s41467-019-09311-w

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533. https://doi.org/10.1038/nature14236

[11] Thomas F. Müller and James Winters. 2018. Compression in cultural evolution: Homogeneity and structure in the emergence and evolution of a large-scale online collaborative art project. *PLOS ONE* 13, 9 (9 2018), e0202019. https://doi.org/10.1371/journal.pone.0202019

[12] Ara Norenzayan, Azim F. Shariff, Will M. Gervais, Aiyana K. Willard, Rita A. McNamara, Edward Slingerland, and Joseph Henrich. 2016. The cultural evolution of prosocial religions. *Behavioral and Brain Sciences* 39 (12 2016), e1. https://doi.org/10.1017/S0140525X14001356

[13] Mancur Olson. 1965. *The Logic of Collective Action: Public Goods and the Theory of Groups.* Harvard University Press, Cambridge, MA.

[14] Christine Payne. 2019. "MuseNet.". openai.com/blog/musenet

[15] Andrea Phillips. 2012. *A Creator's Guide to Transmedia Storytelling: How to Captivate and Engage Audiences across Multiple Platforms.* McGraw-Hill.

[16] Benjamin Grant Purzycki, Coren Apicella, Quentin D Atkinson, Emma Cohen, Rita Anne Mcnamara, Aiyana K Willard, Dimitris Xygalatas, Ara Norenzayan, and Joseph Henrich. 2016. Moralistic gods, supernatural punishment and the expansion of human sociality. *Nature* 530, 7590 (2016), 327–330. https://doi.org/10.1038/nature16980

[17] Peter Richerson and Joe Henrich. 2009. Tribal Social Instincts and the Cultural Evolution of Institutions to Solve Collective Action Problems. *SSRN Electronic Journal* 3, 1 (2009). https://doi.org/10.2139/ssrn.1368756

[18] Peter J. Richerson and Robert Boyd. 1998. The evolution of human ultrasociality. In *Ethnic Conflict and Indoctrination*, I Eibl-Eibesfeldt and F Salter (Eds.). Berghahn, Oxford, 71–95.

[19] Louis B. Rosenberg. 2015. Human Swarms, a real-time method for collective intelligence. In *Proceedings of the European Conference on Artificial Life 2015.* The MIT Press, 658–659. https://doi.org/10.7551/978-0-262-33027-5-ch117

[20] Michael Tomasello. 2009. *Why we cooperate.* MIT Press, Cambridge, MA.

[21] P. Turchin. 2013. The puzzle of human ultrasociality: How did large-scale complexsocieties evolve? In *Cultural Evolution*, PJ Richerson and MH Christiansen (Eds.). MIT Press, Cambridge, MA, Chapter 4, 61–73.

[22] P. Turchin, T. E. Currie, E. A. L. Turner, and S. Gavrilets. 2013. War, space, and the evolution of Old World complex societies. *Proceedings of the National Academy of Sciences* 110, 41 (10 2013), 16384–16389. https://doi.org/10.1073/pnas.1308825110

Appendix

A REPRODUCIBILITY

Data and code to reproduce the analysis are available via Github: https://github.com/chri4354/beeme.

B SUPPORTING FIGURES

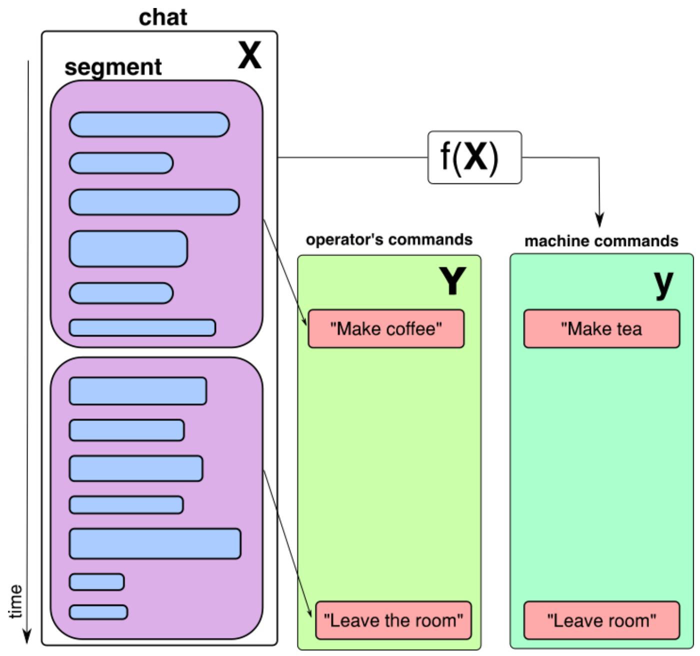


Fig. 5. Data segmentation and algorithmic task. Chat data was segmented based on the commands issued by the operators to the agents. Each chat segment and the associated operator's command represented one data point. Each chat message within the segment was further processed by mapping it into a relevant bigram (not shown). Six algorithms were designed to read in chat logs and output for each segment a (bigram) command. The objective was to maximize the match between human generated and machine generated bigrams.