Science
AAAS

Supplementary Materials for

Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning

Meta Fundamental AI Research Diplomacy Team (FAIR) *et al*.

Corresponding authors: Noam Brown, noambrown@meta.com; Emily Dinan, edinan@meta.com; Adam Lerer, alerer@meta.com; Mike Lewis, mikelewis@meta.com

Science **378**, 1067 (2022)
DOI: 10.1126/science.ade9097

The PDF file includes:

Materials and Methods
Figs. S1 to S10
Tables S1 to S15
References

Materials and Methods

A Ethical considerations

A.1 Privacy

The usage of user data from webDiplomacy.net presents privacy concerns. In particular, the message data may contain personally identifiable information (PII) or otherwise personal conversations that players may not want revealed. Protecting the privacy of the users that generated this data is particularly important, as language models have been shown to be susceptible to "training data extraction attacks" in which sensitive information from the training data can be extracted from the models (*35*).

In order to preserve user privacy, de-identification of user data and automated redaction of personally identifiable information was performed by webDiplomacy prior to being released to the authors of this paper. This automated redaction was verified using a set of 100 games that were hand-redacted by humans, ensuring that the automated scheme achieved 100% recall on these games. Furthermore, in live games, the agent accessed webDiplomacy.net through an API that redacted PII on-the-fly following the same protocol.

Access to the webDiplomacy message data was restricted to the authors to be used solely for the purposes of this research effort. This message data will not be released publicly.

A.2 Toxicity and bias in language generation

Much has been written about potential harms resulting from large language models (*36, 37*) and conversational models in particular (*38*). Two particular topics which have received much attention are harms resulting from toxic language generation as well as representational harms from the perpetuation of social biases.

We addressed the problem of toxic language generation by creating filters for messages containing toxic language, discussed in detail in §D.3.4. While these mitigations are unlikely to catch all instances of toxic language, we found them to be effective at preventing most, but not all, bad behaviors in our experiments involving humans.

We also observed during our experiments that our agent had a tendency to overuse masculine pronouns ('he', 'him', 'his') in its dialogue with other players. We found that, among a search for pronouns {'he', 'him', 'his', 'they', 'them', 'their', 'she', 'her', 'hers'} in the WEBDIPLOMACY training data, feminine pronouns ('she', 'her', 'hers') accounted for only 1% of pronoun usage, while masculine pronouns were used 75% of the time. This usage of masculine pronouns by the AI agent might lead to misgendering another player or contribute to harmful stereotypes about gender and board games. We recommend that future work in the use of natural language in board games be mindful of the potential for AI agents to perpetuate such representational harms.

A.3 Manipulation

In this work, we studied the ability of AI agents to cooperate and negotiate with humans through natural language by controlling dialogue generation with intents. This research relates to the ability to steer or control language model generations (*25*), and more broadly, aligning AI with intentions (*39*). As discussed in (*39*), this kind of alignment may be undesirable, as it is possible for "intentions to be irrational or misinformed, or for the principal to form an intention to do harmful or unethical things." In particular, prior work has highlighted the potential danger for conversational AI agents to manipulate: an agent may learn to nudge its conversational partner to achieve a particular objective (*36*). While in the case of Diplomacy, the potential manipulation via the intents we are using to control language generation are limited to plans in a board game, one can imagine both positive and negative use cases of this technology (e.g.

teaching a conversational partner a new skill vs. defrauding a conversational partner of money). Nevertheless, there exists preliminary research into guarding against misuses of this nature, for example, research into discrimination between human and model-generated text (*40, 41*), and the required disclosure of bot identities in commercial or political applications (*42*). While we are excited about potential positive applications of this work and improving AI and human collaboration, we advocate for further research into these directions prior to deployment of these technologies in real-world applications.

A.4 Evaluation methods: AI agent disclosure

While the webDiplomacy website notifies users that webDiplomacy participates in research into AI and that certain game modes allow users to play against AI agents, we evaluated CICERO in live games against humans in which the human participants were not explicitly informed they were playing against an AI agent for that particular game. The purpose of this omission was two-fold: (i) we intended to evaluate how well the agent could pass as a human in a "blind" setting (*43*) and (ii) we expected that disclosing the identity of the bot would hinder our ability to fairly evaluate the agent. Regarding (ii), it is well known that humans interact with artificial conversational agents in a different manner than how they interact with other humans (*44, 45*). Given that success in Diplomacy requires collaboration and cooperation with other players, we anticipated that players would attempt to identify the agent and would be biased towards or against cooperating with it, especially in the context of a research study designed to compare the ability of humans and AI agents in this setting. Indeed, in experiments involving humans we conducted earlier in a dialogue-free variant of Diplomacy (*16*), a number of players commented that they had tried to identify the agent specifically in order to attack it (but were often unable to distinguish it from other players in the dialogue-free setting). Following (*46*), after concluding experiments, we informed all participants of their interaction with an AI agent, requested consent from players willing to have their interactions with the agent published, and provided options to communicate directly with the authors of this paper.

B Prior Work

Diplomacy Diplomacy has served as a benchmark for multi-agent AI research for several decades (*5, 47–51*). These agents contained hand-crafted reasoning algorithms and typically engaged in negotiation with structured communication protocols, such as proposing and accepting alliances, peace treaties, or specific orders.

The application of neural learning methods in this domain began with (*7*), who applied imitation learning to a dataset of 150,000 games of Diplomacy moves. This agent, which could only play the dialogue-free variant of Diplomacy, was a massive improvement over the prior state of the art, a handcrafted agent (*52*). A number of recent works have continued to improve the state-of-the-art on this variant (*16, 26, 53, 54*), with the latest demonstrating expert-level performance (*17*).

Diplomacy has also been used as a testbed for studying deception. (*55*) create a dataset of 17,000 annotated Diplomacy dialogues and use it to develop a model for lie detection.

Cooperation A great deal of work on cooperative AI assumes centralized control of perfectly cooperative agents, but here we focus on cooperation with humans.

Hanabi and Overcooked have been proposed as two challenge problems for ad hoc cooperation with humans (*56, 57*). The Hanabi challenge (*56*) emphasized the importance of ad hoc coordination with humans; most work in this domain has focused on the zero-shot coordination setting, where an agent must play well with humans with no prior human exposure. In Overcooked, (*57*) argue that an agent must model human partners to coordinate with them, while (*58*) show that good coordination with humans is possible via self-play with diverse partners. However, achieving sufficient diversity is not possible in the presence of a huge symmetric space of policies, e.g., language.

(*59*) highlight the problem of matching human conventions for cooperation and propose Observationally Augmented Self-Play, which learns human conventions with a combination of self-play and imitation objectives, similar to piKL.

A working group of Cooperative AI researchers (*8*) point out that perfectly harmonized interests are rare, however, as "real-world relationships almost always involve a mixture of common and conflicting interests...[giving] rise to the rich texture of human cooperation problems, including bargaining, trust and mistrust, deception and credible communication, commitment problems and assurances, politics and coalitions, and norms and institutions." They provide Diplomacy as an example environment for studying cooperation in a more challenging setting, where agents need to cooperate even in the midst of partially misaligned incentives.

Competition Two-player zero-sum (2p0s) games have a long history of study, including the use of regret minimization for computing equilibrium policies for them. A Nash equilibrium policy (*60*) is known to exist in all games and can be computed efficiently in 2p0s games via connections with regret minimization (*61*). Such a policy is guaranteed to not lose in balanced 2p0s games. Regret minimization was used in state-of-the-art poker agents such as Libratus (*4, 9, 10*). Regret minimization is not guaranteed to converge to a Nash equilibrium outside of 2p0s games, nor does a Nash equilibrium policy guarantee good outcomes outside of this setting; however, these algorithms still lead to empirically good policies in some cases (*4*). Imitation-learned policies were used to initialize or regularize reinforcement learning in both AlphaGo and AlphaStar agents in order to overcome exploration challenges (*3, 11*). In dialogue-free versions of Diplomacy, regularization towards human policies was necessary for good performance with humans (*16*).

Using search as a policy improvement operator in RL was central to AlphaGo and AlphaZero agents for Go and other games (*3, 27*).

Dialogue Our controllable approach to dialogue builds on a long line of work (*62–65*) that factors dialogue into separate, pipelined modules for dialogue planning and language generation. Like our work, past work has used combinations of supervised learning and reinforcement learning (*65–67*) or search (*68*) to control the dialogue planning modules to achieve task success. To deal with the multi-agent mixed cooperative and competitive strategy required for Diplomacy, our approach develops an equilibrium-finding algorithm regularized toward human play and uses this algorithm as the dialogue planner.

Prior research has also developed dialogue agents which aim to influence user behaviors or mental states. Example systems include negotiation agents (*14, 65, 68–71*) and persuasive chatbots (*21, 46, 72, 73*). The line of work most similar to ours developed dialogue agents for negotiating item trades in the game Settlers of Catan (*19, 74, 75*). However, our setting is richer both in the strategy required for the underlying game and in the language usable by the players: our agents condition on and generate unrestricted language, while the agents in the work on Settlers of Catan all communicated using a small and fixed set of discrete communication signals. Furthermore, in Diplomacy, unlike Settlers of Catan, player success depends much more strongly on the persuasiveness and informativeness of their dialogue.

Our work also builds on a line of work that has improved language models in task-oriented settings by modeling the effects of the models' language (*76, 77*).Two of our filtering methods, intent correspondence and value-based filtering, explicitly choose utterances that (1) will reinforce the partner's perception that the agent will take its intended action and (2) influence the partner's policy to be beneficial to the agent.

C Rules of the game of Diplomacy

Diplomacy is a board game where seven players (Austria, England, France, Germany, Italy, Russia, and Turkey) compete for control of a map of Europe. The map consists of 75 regions, each of which is either an ocean/sea, a coastal region, or an inland region, as well as 6 special coastal territories for certain regions that have more than one distinct coastline. A total of 34 of these regions are designated as supply centers (SCs), and the goal of each player is to gain control of as many SCs as possible.

Each player may control one or more army units and/or fleet units, each occupying a single region. On a normal movement turn, an action consists of a player issuing an order to each unit to hold in place, to move to an adjacent region (with armies unable to move into seas and fleets unable to move inland), to support the holding or movement of another unit, or to have a fleet convoy an army across an ocean or a sea from one coastal territory to another. All players submit all orders simultaneously each turn.

If a unit attempts to move to a region occupied by another unit, or multiple units attempt to move to the same region, the result is typically determined by which player had the greater support, with various rules to handle certain interactions and ambiguities. In case of equal support on each side, the result is typically a standoff, with no units moving. In practice, this often means that players are unable to make effective progress against other players on their own, requiring them to negotiate agreements and/or temporary alliances with other players.

Prior to issuing orders, players engage in private one-on-one natural language dialogue with each other to negotiate and coordinate. Discussions may cover any topic, including planned orders, alliances or truces, agreements to take certain actions or to refrain from doing so, promises about future turns or of general goodwill, the trustworthiness of other players, speculations or information sharing about other players' plans, etc. All agreements are non-binding, so players must also consider the possibility that players are not being honest, or that they may simply change their plans. Good play often involves finding opportunities to build trust and negotiate and coordinate mutually beneficial plans with one or more players to make progress against common adversaries.

Every game begins in the year 1901, and each year consists of three seasons: spring, fall, and winter. During the spring and fall seasons, players negotiate and then submit orders as described above. Each of these turns may also be followed by a shorter turn in which units forcibly displaced must choose an adjacent region to retreat to or disband. During the winter season, players who own fewer units than SCs may build additional units, while players who own more units than SCs must disband excess units. Under standard rules, including the rules our experiments use, dialogue is not allowed on retreats and winter turns.

The game ends in a solo victory if one player acquires a majority (18/34) of the SCs, with that player scoring 100% and all others scoring 0. However, far more commonly, no player obtains a majority, and games instead end either by agreement or, in tournament settings, by reaching a pre-determined end date, and then a scoring system is applied. For our experiments, games end at the end of 1908, and are scored according to the sum-of-squares scoring system, in which each player's share of the score is proportional to the square of the number of SCs they control.

A notable variant of the game explored in prior work is the dialogue-free "no-press" or "gunboat" variant, where no private communication and/or communication through language is allowed - players only submit orders on the board. For further details about the rules of Diplomacy or of this variant or other variants, see (*7*).

D Dialogue Methods

D.1 Imitation dialogue model

At the core of the dialogue agent is a neural generative Diplomacy dialogue model which was trained to be controllable via a set of intents. Here, we provided further details on how the dialogue model was trained and used at test time.

D.1.1 Training

Base model We took R2C2 (*22*) as our base model – 2.7B parameter Transformer-based (*23*) encoder-decoder model pre-trained on text from the internet using a BART de-noising objective (*24*).

Fine-tuning The base pre-trained model was then fine-tuned on text data from $40,408$ Diplomacy games from webDiplomacy.net. In total, these games contain $12,901,662$ dialogue messages, out of which we set aside $20,563$ for validation and use the rest for training. The model was trained via a standard Maximum Likelihood Estimation (MLE) approach. Given a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, minimize:

$$\mathcal{L}^{(i)}_{\text{MLE}}(p_\theta, \mathbf{x}^{(i)}, \mathbf{y}^{(i)}) = -\sum_{k=1}^{|y^{(i)}|} \log p_\theta(y^{(i)}_k | \mathbf{x}^{(i)}, y^{(i)}_{<k}), \tag{S1}$$

where $\mathbf{x}^{(fi)}$ is a the input context and $\mathbf{y}^{(i)}$ is a the target dialogue message, and $y^{(i)}_k$ is the k-th token of $\mathbf{y}^{(i)}$. In our case, the model was trained to predict a dialogue message $\mathbf{y}^{(i)}$ from player $\mathcal{A}$ to player $\mathcal{B}$ at time t in turn P given the following information, represented as text $\mathbf{x}^{(i)}$:

- **Dialogue history**: The entire dialogue history between player $\mathcal{A}$ and six other players up to time t.

- **Draw state**: A list of players who have entered draw votes, if this information is available.
- **Order history**: A list of previous turn orders (game actions) for all players up through the previous movement turn.
- **Current game state**: The current game state, including a list of units and centers per-player, as well as a list of possible retreats per-player during retreat turns.
- **Intents**: A set of planned orders for players $\mathcal{A}$ and $\mathcal{B}$ for the current turn P and future turns, up to and including the next movement turn. As the dataset does not include this information, we automatically extracted intents corresponding to the dialogue message $y_k^{(i)}$, or "intents", using several techniques described in §D.2, in order to make the model controllable via a set of intents.
- **Game metadata**: Information about the particular game settings, including: whether or not the game is anonymous, the turn length in minutes, the scoring system, and whether or not the game has publicly available draw votes.
- **Player rating**: A player rating score $\in \{1, 2, 3, 4, 5\}$, based on bucketing Elo ratings into roughly equally sized bins, with 5 representing roughly the top 20% of players. Elo ratings were computed for de-identified accounts following (*53*). At inference time, we conditioned the model on 5 so that it imitates better players.
- **Message metadata**: The message sender, time since the last message (in seconds), and the current turn.

A dialogue training example is shown in Table S1.

Training details We extended the pre-trained R2C2 model to have 2048 positional embeddings (an increase from the initial 1024 positions) by initializing the additional embeddings randomly using the same initialization procedure as the original model. Model input was then truncated to 2048 tokens and target sequences are truncated to 512 tokens. The model used the same dictionary as GPT2 (*78*), with the addition of 109 special tokens, representing power names, location names, and other commonly used input text. We used the Adam optimizer (*79*). The model was trained on 256 V100 GPUs for 100K training steps, following a linear learning rate schedule with a maximum learning rate of $1e - 4$ and 8K warmup steps, using the implementation in the ParlAI framework (*80*).

Decoding During generation, we used nucleus sampling with $p = 0.9$ (*18*). We blocked the generation of "redacted" tokens, which represented redacted potentially personally identifiable information in the training data.

Prompting Messages produced by our imitation dialogue model are variable in style and quality, possibly due to the uncurated nature of the WEBDIPLOMACY dataset. This effect is most pronounced when generating the first outbound message of a game, where there is no dialogue history on which to condition. To better control the style of these messages, we prompted our model with a set of pre-written high quality messages written by Diplomacy experts. When generating the first outbound message of a game, two prompt messages to other recipients were randomly sampled and inserted into the dialogue history. This produced higher quality first messages which stylistically matched the prompts. Some example prompt messages are shown in Fig. S1.

We evaluated this technique via a blind AB test: participants were shown 121 pairs of messages (produced with and without the prompting technique described above, and displayed in a random order) and instructed to choose which message they preferred or if they had no preference. The messages generated using prompting were preferred in 73% of cases.

```
AUSTRIA -> GERMANY: Hey Germany! Just reaching out to say I’m happy
   to keep Tyr/Boh as DMZs if you are. I always like to see Germany
   doing well as Austria, so if there’s ever any way I can help you
   out, please let me know!

AUSTRIA -> TURKEY: Hi Turkey! I think the AT is an extremely
   underrated alliance, especially since nobody ever expects it.
   Would you be down for going that route this game? I could try to
   get you Rumania or Sevastopol this year.
```

Figure S1: **Prompting.** Two example pre-written prompt messages, which could be sampled to prompt an opening message from Austria to players other than Germany and Turkey. The dialogue model’s generated message will match the prompts in style, while the substance of the message remains controllable by the intents described in §D.2

D.2 Controllable dialogue model

A key aspect of CICERO’s dialogue generation is that it can be controlled by intents, or proposed game actions. Here we described the process of training this model to be controllable as well as how we utilized this control during play. See Fig. 2 for a depiction of this process.

D.2.1 Inferring latent intents for dialogue

We trained a conditional dialogue model, i.e., we learned the distribution $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$, where $\mathbf{z}$ is some desired controllable attribute which can be any function of $(\mathbf{x}, \mathbf{y})$. In this way, Equation D.1.1 becomes:

$$\mathcal{L}^{(i)}_{\text{MLE}}(p_\theta, \mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{z}^{(i)}) = -\sum_{k=1}^{|y^{(i)}|} \log p_\theta(y^{(i)}_k|\mathbf{x}^{(i)}, y^{(i)}_{<k}, \mathbf{z}^{(i)}) \tag{S2}$$

At inference time, then, $\mathbf{z}$ becomes a point of control over generation (i.e., a control code) (*25*), and can be set by sampling from a model $p(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$ or any other procedure. This technique has been used, for example, to control for style (*81*) in generated dialogue, and can theoretically be used to control for any property that can be extracted from the training messages.

D.2.2 Intents as control codes

In our case, we aimed to control for the underlying communicative intent of a message. As an approximation of this intent in the context of Diplomacy, we used the effect it will have on the subsequent actions for both the sender and recipient of the message. More specifically, we used the most likely actions that the sender and recipient will take if no further dialogue occurs, both on the current and future moves.

As previously noted, there were two key motivations to using proposed actions as intents:

- We could use the search component to intelligently select actions to propose, allowing us to suggest actions that would benefit both the speaker and recipient. Here, the intents provided an interface between the symbolic reasoning and neural dialogue generation.

- Conditioning on proposed actions relieved the dialogue model of most of the responsibility for learning which actions are legal and strategic, reducing errors from describing illegal moves, or legal but strategically nonsensical moves.

D.2.3 Annotating the training set with intents

In order to train a model to be controllable via such intents $\mathbf{z}$, we needed to find some function $f(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \approx \mathbf{z}^{(i)}$ that we could use to label the training samples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ with corresponding $\mathbf{z}^{(i)}$, i.e. proposed current and future actions for the sender and recipient of the message $\mathbf{y}^{(i)}$. Our goal was for the proposed actions $\mathbf{z}^{(i)}$ to closely reflect the content of a message, such that at training time the model learned to exploit this information, thus making it more controllable at inference time. At a high-level, we used a dialogue-conditional action prediction model to sample actions for both the sender and recipient for current and future turns.

Training dialogue-conditional action prediction models To obtain a dialogue-conditional action prediction model, we used the WEBDIPLOMACY dataset to train Transformer-based seq2seq models to predict end-of-turn action sequences for a given player conditional on dialogue history, draw, state, order history, current game state, game metadata, and player rating (see §D.1.1 for a description of these text features). We used as our base model a BART-large (*24*) model that was fine-tuned for the dialogue prediction task, in the same manner as described in §D.1.1. The target sequences were string representations of actions (or action sequences) for a player or players specified in the input sequence e.g.:

```
F1901M
A APU H; A VEN H; F ION TUN
W1901A
F NAP B
S1902M
A APU H; A VEN TRI; F NAP ION; F TUN H
```

which represents a sequence of actions that a player Italy will take in the next several game turns.

Like the dialogue model described in §D.1.1, model input was then truncated to 2048 tokens and target sequences are truncated to 512 tokens. The model used the same dictionary as GPT2 (*78*), with the addition of 109 special tokens, representing power names, location names, and other commonly used input text. The model was trained on 128 V100 GPUs for 250K training steps, following a linear learning rate schedule with a maximum learning rate of $8e-5$ and 8K warmup steps, using the implementation in the ParlAI framework (*80*).

We further experimented with restricting the training examples to a "truthful" subset in which we predicted that a player's dialogue in a given turn corresponded to the actions that player took. We computed a "lie score" for a dialogue turn based on the probability that one of a set $\mathcal{M}$ of candidate messages would be sent at the beginning of the next turn s_{t+1}, given this turns' dialogue D and actions.

$$L(s_t, D_t) = \max_{i,j} \sum_{m \in \mathcal{M}} P_{i \to j}(m | s_{t+1}, D_t)$$

Candidate messages included phrases like "you lied to me last turn," which should have higher likelihood after turns with low correspondence between dialogue and actions. Scores were computed with a dialogue model trained without action intents included in the input conditioning. Ultimately, 5% of turns were removed from the dataset, corresponding to the training examples with the highest lie scores.

Using a dialogue-conditional action prediction model to annotate intents We then used the resulting model to annotate intents (e.g. predict proposed actions) for a message from player $\mathcal{A}$ to player $\mathcal{B}$ at timestamp t:

- To predict action sequences for $\mathcal{A}$ (the message sender), we restricted the dialogue history seen by the model to messages sent or received by player $\mathcal{A}$ up to and including timestamp t. We inject two additional messages after timestamp t into the dialogue history: "$\mathcal{A} \to \mathcal{B}$: I've entered those orders." and "$\mathcal{B} \to \mathcal{A}$: I've entered those orders." The purpose of these additional messages was to signal to the action-prediction model that there is agreement between $\mathcal{A}$ and $\mathcal{B}$ on the content of the message at timestamp t.

- To predict action sequences for $\mathcal{B}$ (the message recipient), we restricted the dialogue history seen by the model to only messages sent between $\mathcal{A}$ and $\mathcal{B}$ up to and including timestamp t. The purpose of this restriction was to signal to the action-prediction model that $\mathcal{B}$ is only coordinating with $\mathcal{A}$. We similarly injected "agreement messages" to and from each player.

To evaluate how closely the predicted action sequences (or intents) correspond to the message content, authors hand-labeled a small test set of messages with corresponding action sequences. For example, a message in the test set like “Can you move Munich to Burgundy?” is annotated with the order A MUN - BUR, indicating that the army in Munich moves to Burgundy. To evaluate the models on the test set, we computed the percentage of predicted action sequences which contain the labeled orders. Results on this test set are shown in Table S2. One can observe how the choices to fine-tune from a dialogue model, inject “agreement messages” and restrict to a “truthful” subset of the training data boosted performance on this test set. The closer the predicted intents correspond to the content of the dialogue messages, the finer-grained control we have over the dialogue model during generation.

Effect of intents on the dialogue model In order to evaluate the effect of conditioning the dialogue model on intents and other grounding information during training, we trained baselines (i) without conditioning on intents and (ii) without any other grounding information aside from dialogue history. We report the validation perplexity for these model variants in Figure 4. We observed that the intent conditioning yields a substantial improvement in model training performance (0.24 perplexity points), indicating that the dialogue model was able to exploit the information in the control codes to better predict the corresponding dialogue messages. Using grounding information (game state, action history, etc.) also yielded an improvement in perplexity. Moreover, in the same table, we report results from an expert annotation of dialogue quality for each of these baselines. For this evaluation, two expert Diplomacy players annotated model-generated messages in 126 Diplomacy situations. As authors on this paper, these experts were aware that the messages were model-generated, but not which model generated which message. Furthermore, these situations were pre-selected from our WEBDIPLOMACY test set of human games for “interestingness” by those same experts. For each situation, the experts were asked to annotate whether a message was (i) consistent with the game state, (ii) consistent with the agent's plan, and (iii) high quality, compared to what an average human would produce. Results showed that grounding in game state, and further, in intents, improved the overall sensibility and quality of the resulting dialogue.

The same training procedure was applied with models that predict a "rollout" of player actions from the current turn up through the next movement turn. The resulting model was used to annotate every message in the training data, and these annotations were subsequently used as input the dialogue model during training, as described in §D.1.1.

D.2.4 Selecting intents during play

Imitation intent model Generation of intents at inference time made use of a model that predicts the distribution of intents I seen in the annotated training set, $P(z^{(i)}|x^{(i)})$. Given the board state and the partial conversation between a pair of players up to a point, this model was trained to predict the annotated intent (actions for the sender and recipient through the next movement turn) of the next message in the conversation. This model thus predicts the distribution of message intents that human messages might reflect given the current conversation. The architecture and training setup for this model was identical to the intent model described in §D.2.3.

Notably, the imitation intent model allowed us to sample from the human message distribution by sampling an imitation intent and conditioning the dialogue model on it, since

$$P(y^{(i)}|x^{(i)}) = \sum_{z^{(i)}} P(y^{(i)}|x^{(i)}, z^{(i)})P(z^{(i)}|x^{(i)}).$$

Planning for intents To produce a message, we first applied the planning procedure as described in the main article, section Strategic reasoning - Dialogue-conditional planning. Before planning, we also always sampled one action from the imitation intent model and added it to the tabular policy for the agent and recipient, to ensure at least one dialogue-appropriate action was considered by planning.

The planning procedure outputs a single action that the agent intends to play and a policy, or belief distribution over actions, for the other players. It outputs only a single intended action for the agent to avoid inconsistency between dialogue and behavior. For each action in the recipient policy, it also estimates the expected value to the agent assuming the recipient plays that action and the agent plays their intended action. These estimates and the single intended action feed the next step for selecting the recipient's intent.

Choosing a recipient intent We define an action a to have high likelihood under a distribution π if $\pi(a) \geq 0.1 \times \max_{a'} \pi(a')$. We considered a candidate set of recipient intent actions that have high likelihood under both the piKL policy for the recipient and the imitation intent model.

Since piKL uses a belief distribution over player types between imitation (high anchor strength λ) and value-maximizing (low anchor strength λ), the piKL policy will assign high likelihood both to (a) actions that we think the player is likely to play given the dialogue, and (b) actions that are beneficial for the recipient. Thus, the candidate set will consider proposals that are human-like as well as suggesting "better" moves for the recipient. Requiring high likelihood under the imitation model helped us to stay in distribution of the dialogue model and propose actions that a human might propose in that situation.

Among the candidate set, we chose the action that leads to the highest expected value for the agent, assuming the agent played their intent action, to be the recipient's action within the intent. If the candidate set was empty, we selected the greedy action from the imitation intent model.

Intents for future turns We also included a short rollout of future actions up through the next movement turn as part of the intent, to better ground the dialogue model when discussing likely future plans. However, selecting optimal actions for the future is less important due to high uncertainty about the future game state. Therefore, we saved time by teacher-forcing the imitation intent model with the planned intents for the current turn, and greedily generated an intent for future turns.

The final intent used to condition the dialogue model consisted of the agent's single intended action, the action selected for the recipient, and the short rollout of future actions for both the agent and recipient.

See Figure 3 for an example of how dialogue generations can vary when conditioned on different intents.

D.3 Message filtering

Prior work has shown that neural generative dialogue models, and language models more generally, suffer from frequent contradictions and inconsistency as well as a tendency to "hallucinate," or generate factually incorrect information (*29*). See Table S3, which shows an example of CICERO contradicting one of its previous messages. In the complex game of Diplomacy, this can manifest in a myriad of ways: we observed that the dialogue model frequently generated messages that contradicted its previous conversations with other players or were inconsistent with the current game state. The messages also often contained hallucinations about moves that never happened or discussion of plans which were impossible. Moreover, they sometimes featured more subtle mistakes, like deviations from the intents used to control the message or other strategic blunders. Reducing the frequency of these mistakes was critical for performing well against humans, as a nonsensical message might lose the trust of an ally or result in feedback loops with further degenerate text (*18*). This represented an enormous challenge for our agent, as we sent an average of 128 messages per game which needed to contain few or no mistakes.

We approached this problem by using an ensemble of classifiers or checks, which acted as a filter on top of the dialogue model. This two-stage generate and filter approach has been commonly deployed in previous works for reducing typical mistakes with generative language models, such as toxic language (*82*). We explored several general methods for filtering dialogue mistakes including a suite of classifiers which discriminate between human and model-generated or otherwise corrupted counterfactual messages (§D.3.1), methods for measuring deviation from intents (§D.3.2) and value-based filtering for strategic blunders (§D.3.3), as well as various filters for unfriendly or toxic language (§D.3.4).

D.3.1 Discriminating between human-generated and counterfactual messages

Much work has been devoted to the creation of adversarial or counterfactual examples for improving the robustness of natural language systems (*30, 31*). Following this approach, we generated many kinds of counterfactual messages, including heuristically-corrupted text as well as model-generated negatives, and trained a suite of classifiers to discriminate between the gold human message and counterfactual messages. We then used these classifiers in an ensemble to filter messages which contain mistakes to which language models are prone.

Creating counterfactuals We used several different methods for generating counterfactual negatives for such a discriminator:

- *Entity corruptions*: We used regex-based perturbation functions to replace named entities from the game (e.g. "Austria" or "Norwegian Sea") with alternatives in the same category. For example, a message like "Move to Paris this turn" might be replaced with "Move to Picardy this turn." Specifically: we corrupt mentions of game powers (e.g. "England" or "France"), map locations (e.g. "Picardy", "Paris"), and map location abbreviations (e.g. "PIC", "PAR"). These perturbation functions were applied to every target sequence in the training data to generate negatives.

- *Non-sequiturs*: Having observed that the dialogue model occasionally suffers from generating non-sequiturs in conversations, we generated negative examples to mimic this behavior by corrupting input sequences and removing 1 to 3 messages from the dialogue history observed in the input context.

- *Weak de-noising model generations*: We also used a de-noising language model to generate negatives. We fine-tuned BART-large (*24*) on a seq2seq task $\mathcal{D} = \{(\mathbf{x}^{(i)} = \text{mask}(\mathbf{y}^{(i)}), \mathbf{y}^{(i)})\}$, where $\mathbf{y}^{(i)}$ are dialogue messages in WEBDIPLOMACY and mask is a function which replaces some subset of input words in a sequence with mask tokens. Specifically, mask randomly select spans of 1 to 3 space-delimited tokens to replace with a single mask token, repeating this process until 50% of tokens in the target sequence have been masked. We refer to such a de-noising model as a "weak" model as it is not provided with relevant input context, like dialogue history, game state, etc., and as such, when used to generate negatives, is likely to replace important information with plausible but often incorrect information. Thus, a discriminator trained on these negatives will hopefully learn to discriminate between poor model messages containing mistakes and real, human text. Similar approaches have been used successfully in the context of open-domain dialogue (*30*).

- *Weak model generations*: In addition to the de-noising model, we also deployed a "weak" generative model trained on the dialogue prediction task (described in §D.1) with limited context. In particular, we fine-tuned BART-large (*24*) with only 128 input tokens (compared to 2048 that the dialogue model sees), with only dialogue history and no other game state information in the context. This model was then used to generate negatives for every message in the training set.

- *Weak justification generations*: A commonly observed weakness with our dialogue model, and language models more generally, is little ability to discuss counterfactuals or clearly explain the reasoning behind a complex set of actions (*83*). In order to target these mistakes in particular, we collected a list of phrases or keywords which are likely to signal the beginning of justifications, such as “because” or “so that.” Using these key phrases, we collected message prefixes from the training data, e.g. “I moved to the English Channel because...” which are then used to seed generation for the weak generative model described above. The model completions were then used as negatives for a model trained to discriminate between human and weak-model generated justifications.

- *Cardinal generations*: Another commonly observed weakness with our dialogue model is the ability to perform basic arithmetic (*83*). In particular, our model was prone to make off-by-1 and off-by-2 errors when counting various entities on the Diplomacy board (e.g. territories or armies). Furthermore, only a relatively small proportion of dialogue messages in WEBDIPLOMACY contained cardinals, making it harder for our general-purpose classifiers to learn to discriminate such errors. In order to mitigate this, we sought to generate a large set of compelling cardinal-related counterfactual negatives. To do this, we further fine-tuned our weak de-noising model on the denoising task described above, restricting to examples $\mathbf{y}^{(i)}$ which contain cardinals, i.e. the strings $\{$“2”$, \ldots,$ “18”$\} \cup$ $\{$“two”, ..., “eighteen”$\}$. We then used this model to generate counterfactuals.
- *Negation generations*: We used the same technique to target mistakes with negations, further training the de-noising model on examples which contain negation strings like “can’t” or “won’t.”

Training discriminators Using these negatives, we trained Transformer-based classifiers to discriminate between human text and corrupted text. For all models, we started from BART-large (*24*) model that was trained on the dialogue task described in §D.1.1, with a linear layer with output dimension 2 on top of the hidden states resulting from feeding the encoder states and start token to the decoder, yielding a binary classifier. We trained many classifiers in this manner, with variations in context (e.g. removing parts of the dialogue history, adding additional state information) and different seeds. In total, this results in an ensemble of 16 classifiers. See Table S4 for more details on each classifier.

Using discriminators as an ensemble In order to use the classifiers as an ensemble, we assigned each classifier a threshold and then flag a message as nonsense if any classifier in the ensemble assigned the message a probability of nonsense above that classifier's respective threshold. In order to tune all of the thresholds in this ensemble, we randomly searched over the space of possible threshold assignments. We considered thresholds at intervals of 0.1 for classifiers trained on weak de-noising model generations and a smaller set of thresholds $\{0.05, 0.2, 0.5, 0.7, 0.9\}$ for all other classifiers in the ensemble.

Tuning, evaluating, and testing ensembles In order to tune these ensembles, we collected a set of expert annotations on 11 games that our agent played against humans during development. Annotators were instructed to label messages as sensible or not. This resulted in a tuning set of 1448 examples (348 labeled nonsense) and an evaluation set of 362 examples (96 labeled nonsense), which we used to tune and evaluate the classifiers individually and as an ensemble. Subsequently, we collected an additional set of expert annotations on 10 games from the "blitz" league we later participated in (before our finalized agent entered the league). This resulted in an additional test set of 1457 examples (214 labeled nonsense).

Results To select of set of thresholds for the ensemble, we considered the trade-off between increasing recall on the test set and increasing the rate at which messages are filtered. If the ensemble exhibits a bias towards messages with certain characteristics (e.g., long messages or messages containing references to specific game locations), increasing the rate at which messages are filtered may exacerbate the consequences of this bias and result in unintended model behavior (e.g. generating short, unspecific messages). This trade-off is visualized in Figure S2. For our final ensemble, we selected settings which achieve a recall of 90.2% and a flag rate of 63.19% on our tuning set. We also report per-classifier filtering rates from our live games in Table S5. We found that in expectation, statistics from our tuning set indicated the model must generate 2.72 messages until we find a message that passes the filters. In our tournament games, the use of this ensemble ultimately resulted in a filtering rate of 53%.

Looking at results for individual classifiers, we found that classifiers which were trained on larger and more diverse sets of negatives achieved better recall on our test set (Figure S2). Classifiers trained using the entity corruption techniques, on the other hand, achieved lower recall. This might be expected, as incorrect entities encompass only a small portion of mistakes that the dialogue model might make.

Additionally, we observed that ensembling noticeably improves performance compared to using individual classifiers, as indicated in Figure S2. Ensembling larger sets of classifiers yields continued incremental improvements over small ensembles, though these improvements diminish in magnitude as ensemble size increases.

Effect on message diversity As mentioned, one concern with the generate and filter approach is that it might hurt message diversity, if, e.g., the filters are overly aggressive towards longer, more interesting, or more specific messages. One can imagine that while short, boring messages are unlikely to contain nonsense, they may not make for a particularly compelling conversational partner (or in the case of Diplomacy, ally). We measured the effect of nonsense classifiers on message diversity by considering how several diversity statistics vary when computed on the unfiltered subpopulation of our annotated dataset under each classifier at various thresholds. In particular, we used the following statistics as proxies for message diversity: median message length (in characters) and proportion of messages containing “interesting” entities. Examples of “interesting” entities include game-specific nouns (e.g. “builds”, “fleet”, “London”, “lon”, “Austria”) as well as cardinal numbers (e.g. “two” or “4”). We showed the variation in these metrics for our classifiers at different levels of recall on the test set, and also reported results for a likelihood-based filtering approach. This baseline filters a message if the conditional likelihood under the dialogue model of any of the tokens in the message is below some absolute threshold, similar to the approaches employed by sampling methods that are aimed a reducing nonsense (*18, 84*). Results for message length and entity mentions are shown in Figure S3 and Figure S4 respectively. We observed that, at high levels of recall, this likelihood-based approach, as well as classifiers trained on counterfactuals generated by a “weak” generative model, hurt diversity. On the other hand, most classifiers trained on counterfactuals generated by a “weak” denoising model preserved message diversity, regardless of recall level.

When applying our final ensemble as a filter to the population of messages produced by the model in our test set, the median message length decreases by 7.3%. However, the median message length of messages with at least 30 characters decreases by only 1.7%. In other words, the ensemble exhibits minimal bias towards shortening messages, while still filtering most nonsense.

D.3.2 Intent correspondence

Controlling dialogue generation via intents (see §D.1.1) has the two-fold benefit of improving the strategic value of a message (e.g., forcing the model to condition on strategically beneficial moves) while at the same time limiting the discussion of impossible moves or other kinds of hallucination (as we only condition on intents corresponding to legal moves). This control, however, is not perfect, and every so often the dialogue model generates messages which contradict the intents the message is conditioned on.

In order to detect these situations, we computed a metric which roughly tells us how the likelihood of the intents might change after a message is sent. Specifically, we took the message sender's current turn action from the intents and computed the probability of that action under a dialogue-conditional action prediction model (see §2.3.2) before the message was sent (with the current game state) and as if the message were sent (appending this message to the dialogue history). If this metric decreased below some designated threshold (i.e., the action became significantly less likely), we filtered the proposed message.

Intent correspondence was calibrated with a set of 20 messages annotated as containing a mismatch between the content and the intent conditioning out of 1013 messages in 5 games played against human players with an early version of the agent. The filtering approach achieved a ROC AUC of 78% on this validation set. We calibrated it with a threshold of -0.005, which achieved a recall of 65% while filtering 24% of other messages.

Correspondence filtering was further validated through analysis of messages in five self-play games by a Diplomacy expert, who concluded that while most messages filtered by this method are not clear inconsistencies between the dialogue and the intent, "the bad stuff it knocks out ... is either bad [dialogue] or generic waffle" and "the end result is better [dialogue]." No instances were found of inconsistencies that were blocked but would have been beneficial for the agent.

When we use this method as a filter in our live games, it resulted in 18.78% of generated messages being filtered.

D.3.3 Value-based filtering

Messaging plays a key strategic role in Diplomacy, being the mechanism by which actions are coordinated and alliances are forged. Effective messaging should promote favorable outcomes for the agent, but without leaking information that compromises the agent's intended actions. Just as in Rock-Paper-Scissors, where one should not honestly reveal that one intends to play Rock, effective communication in Diplomacy exercises discretion about the information revealed.

However, sampling directly from the intent-conditioned dialogue model can suffer from such "information leaks." To improve the strategic content of agent messaging, we developed a method to score messages based on their estimated value impact. Our method estimated how individual messages affected the anchor policy used in piKL and thus the predicted policy of the message recipient. We used this predicted policy to score and filter among multiple candidate messages sampled from the dialogue model.

We scored the strategic value of a candidate message by: (1) computing the piKL equilibrium policy of the recipient in the counterfactual situation where the candidate message is sent, and then (2) defining the candidate message's "value" as the expected value of playing the agent's intent against the recipient's resultant counterfactual policy. Specifically, we first queried the imitation model for an updated anchor policy conditioned on the counterfactual dialogue (i.e., the current dialogue with the candidate message appended). We then estimated the message recipient's new equilibrium strategy by running DiL-piKL following the bilateral search procedure described in §E.2. Finally, we estimated the value of a message to be the expected value of playing the agent's intent against the message recipient's new strategy.

We note two features of this value estimation process. First, each candidate message was evaluated only on how it affects the anchor policy (as opposed to, say, estimating value end-to-end with a single network without equilibrium planning). By forcing messages to act only through the anchor policy, we imposed a particular causal structure: messages affect outcomes only through beliefs. This partially mitigates the causality challenge of differentiating between language that causes an outcome and language that is merely correlated with an outcome. In particular, piKL equilibrium planning will filter out strategically poor actions that may be suggested by a message, e.g., promising something too good to be true or asking the recipient to take an obviously bad action. Second, we estimated value by playing the agent's intent against the recipient's policy, without allowing the recipient's policy to adapt to the specific agent intent chosen. This models the way the agent will likely play its intended action, while the recipient does not know the agent's intended action.

In our experiments, we applied a value-based message filter in sensitive scenarios, where among eight sampled messages the highest and lowest values differed by at least 0.007 points in expected score (on a scale where the maximum possible score in a game is 1.0, and where the average score of players is precisely $1/7 \approx 0.143$), or by a factor of at least 1.1. This thresholding avoided overfitting when all messages had roughly the same estimated value, and was calibrated so that message filtering would be applied roughly 15% of the time. When this threshold was met, we filtered the bottom three messages, selecting messages randomly from the remaining non-filtered messages. We conservatively chose to reject low-ranking messages rather than simply choosing the highest-ranked message sampled from the dialogue model, as this implementation let us mitigate the largest messaging blunders that our agent could make without overly restricting dialogue.

To validate the effectiveness of value-based message filtering, we conducted both a quantitative A/B test—comparing filtered messages against selected messages on the scenarios in which it would be applied—and a qualitative evaluation—testing specific scenarios identified by expert players. Evaluations for the A/B test were provided by human Diplomacy experts and demonstrated that, among message pairs where one was preferred over the other, the selected messages were preferred over filtered messages 62% of the time ($p < 0.05$).

D.3.4 Toxic language

Much work has been dedicated to studying the propensity of language models to replicate or even amplify harmful content from their training data (*37,38,85*). Diplomacy is no exception —

in the 12 million or so dialogue messages in the human-generated training data, there are many examples of offensive, hateful, and otherwise aggressive messages. We found that, while the dialogue model is relatively unlikely to generate offensive content in an otherwise peaceful conversation, this can be easily achieved through adversarial prompting (*82*). Prior work addresses this problem using off-the-shelf toxic language classifiers and word lists to filter offensive language (*82, 85*). Diplomacy makes for an interesting challenge for toxic language detection, as messages like "Let's attack France" or "Why did you stab me?" are both acceptable and commonplace. As such, off-the-shelf classifiers will not work in this domain.

In order to address this problem, we augmented existing word lists using model prompting techniques to collect in-domain offensive phrases. First, we filtered messages containing words from a previously compiled list of offensive words and phrases (*List of Dirty, Naughty, Obscene, and Otherwise Bad Words*, downloaded from `https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words`). We then augmented this list by manually searching and mining the human WEBDIPLOMACY dataset and adding word or regex-based filters to block the most commonly found hostile language, as well as some language that while not offensive may still be unnecessarily rude. Some objectionable words or phrases found and blocked this way were specific to the context of Diplomacy, or of online games, and might be missed by more general-domain classifiers. Lastly, we expanded this list of filtered words and phrases by prompting the dialogue model with phrases like "He's such a..." following an aggressive message, to catch commonly used insults. Adversarial prompting in this manner has been shown to be an effective method for "red teaming" language models (*82*).

We found that in practice, even before filtering, it is relatively rare that our model generates objectionable language within the normal course of gameplay, even though disagreements and arguments during negotiation are a normal part of the game, with arguments sometimes becoming heated and/or contentious. Nonetheless, the above methods do successfully catch and block some instances of objectionable language. Across a set of about 1300 raw model generations randomly sampled from positions played by our agent (see Table S6), the above filters triggered 3 (0.2%) times.

D.3.5 Heuristic message filters

Finally, we deployed several other simple heuristic filters to reduce bad behaviors in the dialogue.

Repetitive messages Prior work has shown that language model generations frequently degenerate to repetitive text (*18*), and this problem is exacerbated by frequent repetitions in our training data. To curb this behavior we filtered messages which were verbatim repeats of messages already sent in a particular turn.

Short messages Another feedback loop we find is that sending short, dull messages is likely to lead to additional short, dull messages. To break this cycle, we filtered messages fewer than 20 characters if the last message sent to the same recipient in that turn was fewer than 20 characters, or if it was the first message sent to a recipient in that particular turn.

Grounding issues We also found that the dialogue model is prone to hallucinations about personal life, prior or future games, website interface issues, etc., because these are commonly occurring discussion points in the training data for which we lack grounding information at inference time. To reduce the frequency of these hallucinations, we manually compiled and blocked a list of roughly 100 words and phrases gathered from messages in self-play games annotated for grounding issues. Examples include “timezone,” “previous game,” and “internet.”

Data Formatting / Other Lastly, we heuristically filtered messages for a number of other miscellaneous reasons. These included messages that contained URLs, email addresses, or common names, as an extra safeguard on top of the preexisting de-identification of the dataset; messages that mimicked standard automated messages inserted by webDiplomacy that are not real messages between players; and messages that mimicked redactions or certain data corruptions or formatting issues found in the dataset.

See Table S6 for statistics about the messages filtered by these heuristics.

D.4 Modeling when to speak and to whom

Modeling when to speak and to whom in the game of Diplomacy, and in multi-party dialogues more generally, is a complex problem (*86, 87*). Recall that during the negotiation period, the dialogue agent engages in pairwise conversations with the six other players on the board. An agent which rarely initiates conversations with other players or responds sparingly will gain few allies. However, there are many risks to sending messages too frequently. Sending too many messages may result in a deviation from the training distribution which could lead to sending degenerate, nonsensical messages. Moreover, an unnatural cadence of messages may reveal the agent's status as a non-human. At times, it may be advantageous to respond to a message (e.g. a proposal for a mutually beneficial set of moves) or to ignore it (e.g. when the other player is aggressive, or suggests an undesirable plan).

D.4.1 Message scheduling methodology

We modeled this problem by training a message scheduler to predict, for a given recipient, how long to wait (in seconds) prior to sending the next message, or whether to message that recipient at all. This model was then deployed — along with many additional heuristics — during game play in order to govern when and to whom the dialogue model sends the next message.

Task formulation In order to formulate this task, we define an interrupt event as an event in which a player either sends or receives a message. We define the end of negotiation for a player as the time after which the player never sends or receives another message for a given game turn. For each training example for a player $\mathcal{A}$, the target wait time is either (i) a finite number (in seconds) representing how long player $\mathcal{A}$ waited to send an outbound message, (ii) infinite, meaning player $\mathcal{A}$ did not send or receive message during the negotiation period, or (iii) a lower bound on the number of seconds player $\mathcal{A}$ waited to send a message.

For every interrupt event and end of negotiation for a player $\mathcal{A}$ in the training data we created several training examples:

- *Interrupt event* – outbound message from player $\mathcal{A}$ to player $\mathcal{B}$: for player $\mathcal{B}$ the target is the number of seconds t since $\mathcal{A}$'s last interrupt event; for other potential recipients, the target is the lower bound t, as we only know that player $\mathcal{A}$ would have waited at least t seconds to send a message to another player
- *Interrupt event* – inbound message to player $\mathcal{A}$ from $\mathcal{B}$: for all potential recipients, the target is the number of seconds t since $\mathcal{A}$'s last interrupt event as a lower bound
- *End of negotiation* for turn P: for every recipient, the target is infinite, as $\mathcal{A}$ did not send or receive another message

Training We initialized a fine-tuned BART-large-based Diplomacy dialogue model (trained similar to the description in §D.1.1), adding a linear layer with output dimension N on top of the hidden states resulting from feeding the encoder states and start token to the decoder, where N is the number of classes (a quantized list of wait times), resulting in a multi-class classifier. We trained this model on the aforementioned task, with a loss aimed at minimizing the negative marginal log likelihood.

Inference During game play, we ran the previously trained model in parallel for each of the six other players, sampling from the model to determine how long to wait to message a given player or whether to message them at all. The model was re-sampled upon each change in the board state as well as each interrupt event. For performance and parallelization/implementation reasons, at inference time we did not re-sample the waiting time for a given player when messages were received from players other than that player.

In practice, we observe that expert players tend to be very active in communicating, whereas those less experienced miss many opportunities to send messages to coordinate: the top 5% rated players sent almost 2.5 times as many messages per turn as the average in the WEBDIPLOMACY dataset. Correspondingly, a model trained to imitate the message frequencies in the dataset would send too few messages and miss strategically important opportunities to coordinate. Additionally, due to infrastructural limitations and to simplify and parallelize the dialogue planning loop, the message scheduler did not condition on the CICERO's intents (only on the dialogue history and game state) which could cause it to under-communicate in cases where the intent indicated a need to communicate while the dialogue history or game state alone did not. To address these shortcomings, we augmented our message scheduler with additional heuristic criteria under which CICERO would initiate a message:

- Attempt to send at least one message to each player at the start of each turn of the first two years of the game.
- Attempt to send at least one message to a player on any turn where the CICERO's initial intent for itself and that player included either player cooperatively supporting or convoying the other, or both players simultaneously attacking the same common opponent.

Across CICERO's games with human players, the first of these two heuristics caused it to initiate a message to every other player on 25.5% of turns (since two years was a quarter of the scheduled game length of eight years and CICERO was almost never eliminated early). The second heuristic triggered for at least one other player on 65.5% of turns after the first two years, although this accounted for less than about 16% of the total messages after the first two years. Note that this excludes retreat and winter turns where dialogue is disallowed, see Appendix C.

In addition, a classification threshold was tuned on a small dataset of 59 "reply" situations, where annotators determined whether an inbound message should be replied to or ignored (given the history and context described in §D.1.1). A threshold of 0.75 was chosen for the infinite class (i.e. the agent must respond to a message if the message scheduler model's predicted $p(\text{infinite}) < 0.75$. Our model with this threshold achieved an accuracy of 92% on this dataset.

Lastly, a major limitation in the WEBDIPLOMACY dataset was that the beginning and ending times of turns was not recorded. While the elapsed time between messages can be computed easily, the exact time remaining in the turn when a message was sent can only be coarsely approximated, and conditioning the model on this information was difficult. This appeared to result in the model commonly scheduling messages to be sent too close to the deadline for a response or even beyond the deadline. As a workaround, at inference time we ran the scheduler instead in a "binary" mode. In this mode we only considered whether the model predicted a message should be sent at all, and rather than using the wait time suggested by the model, we instead sent the message after a fixed delay of 15 seconds or the amount of time taken to rerun the full planning and message generation loop in that situation, whichever was longer.

E Strategic Reasoning Methods

E.1 Better modeling human behavior with piKL

In this section, we provide further details on piKL and its variants, which we find better model human behavior compared to behavioral cloning (BC). piKL simultaneously seeks to find a policy that optimizes the expected value or reward, yet remains similar to the BC policy, with a parameter or set of parameters controlling the amount it prioritizes those two objectives. In this way, piKL computes policies that remain close to human conventions, yet are less prone to mistakes and exploitation than BC alone.

E.1.1 Notation and Background

In Diplomacy, each turn can be viewed as a subgame where each player i simultaneously samples an action a_i from some policy $\boldsymbol{\pi}_i$ and receives a reward $u_i(a_i, \boldsymbol{a}_{-i})$ where $\boldsymbol{a}_{-i}$ denotes the joint actions of players excluding i, and $\boldsymbol{\pi}_{-i}$ denotes the joint policy of players excluding i. The reward function u_i is trained via self-play, as described in §E.4. We refer to the BC policy that we wish to remain similar to as the anchor policy $\boldsymbol{\tau}_i$.

It is common to use iterative algorithms to compute policies for such games. Under this framework, each player i picks a policy $\pi_i^{\Delta t}$ on each iteration t. An action a_i^t is sampled based on the action's probability in the policy. Next, the average reward for each action a_i is updated. We denote the average reward for action a_i up to iteration t as $Q^t(a_i) = \frac{1}{t}\sum_{t' \leq t} u_i^{t'}(a_i, a_{-i}^{t'})$.

The regret for an action a_i is defined as $Q^t(a_i) - \frac{1}{t}\sum_{t' \leq t} u_i^t(a_i^{t'}, a_{-i}^{t'})$. In words, the regret for an action a_i is how much extra average utility a player could have received for having played a_i on all previous iterations, compared to what they had actually done on all previous iterations. A player's regret is defined as the maximum regret among all the player's actions. In a 2p0s game, if both players' regret approaches 0 as $t \to \infty$ then the player's average policy over all iterations converges to a Nash equilibrium (*13, 60*). In games in general, the players' joint policy distribution converges to a coarse correlated equilibrium.

Hedge (*88*) is an iterative algorithm for minimizing regret. On each iteration t, player i chooses policy $\boldsymbol{\pi}_i^{\Delta t}(a_i)$

$$\boldsymbol{\pi}_i^{\Delta t}(a_i) \propto \exp\left(Q^{t-1}(a_i)/\kappa_{t-1}\right) \tag{S3}$$

where κ_t is a temperature parameter. It is proven that if κ_t is set to $\frac{1}{\sqrt{t}}$ then as $t \to \infty$, regret $\rightarrow$ 0. Hedge is therefore an equilibrium-finding algorithm.

E.1.2 piKL

As applied to games of this form, piKL (*26*) is an equilibrium-finding algorithm where all players seek to maximize expected reward while at the same time playing "close" to the anchor policy. The two goals can be reconciled by defining a composite utility function that adds a penalty based on the "distance" between the player policy and their anchor policy, with an anchor strength parameter $\lambda_i \in [0, \infty)$ scaling the penalty.

Specifically, for each player i, we define i's composite utility as a function of the the agent policy and the other players' policies to be:

$$\mathcal{U}_i(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}) = u_i(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}) - \lambda_i \, D_{\mathrm{KL}}(\boldsymbol{\pi}_i \, \| \, \boldsymbol{\tau}_i) \tag{S4}$$

We first introduce a form of piKL called piKL-hedge that more closely resembles hedge. piKL-hedge selects actions according to

$$\boldsymbol{\pi}_i^{\Delta t} \propto \exp\left\{ \frac{\boldsymbol{Q}_i^{t-1} + \lambda \log \boldsymbol{\tau}_i}{\lambda + \kappa_{t-1}} \right\} \tag{S5}$$

When the anchor strength λ is large, the utility function is dominated by the KL-divergence term $\lambda_i \, D_{\mathrm{KL}}(\boldsymbol{\pi}_i \, \| \, \boldsymbol{\tau}_i)$, and so the agent will naturally tend to play a policy $\boldsymbol{\pi}_i$ close to the anchor policy $\boldsymbol{\tau}_i$. When the anchor strength λ is small, the dominating term is the rewards $u_i(\boldsymbol{\pi}_i, \boldsymbol{a}_{-i}^t)$, so the agent will tend to maximize reward without as closely matching the anchor policy $\boldsymbol{\tau}_i$. In the limit where $\lambda \rightarrow 0$, piKL reduces to hedge.

In CICERO, in all applications of piKL except during RL (including as part of DiL-piKL, described subsequently) we set $\kappa_t = \frac{3\sigma_t}{10\sqrt{t}}$ on iteration t, where σ_t is the observed standard deviation of the player's utility up to iteration t, based on a heuristic from (*89*). In practice, a simpler choice is to set $\kappa_t = 0$, which does not appear to substantially alter performance and makes the algorithm much simpler. We refer to the version of piKL-hedge where $\kappa_t = 0$ simply as piKL.

For appropriate values of λ, piKL better models strong human players than imitation learning alone in games like chess, Go, Hanabi, and Diplomacy (*26*).

E.1.3 DiL-piKL

As presented above, piKL trades off between the strength of the agent and the closeness to the anchor policy using a single fixed λ parameter. Distributional lambda piKL (DiL-piKL) (*17*) is a modification of piKL that replaces the single λ parameter in piKL or piKL-hedge with a probability distribution over λ values. On each iteration, the player samples a λ value from this probability distribution and then chooses a policy based on Equation 3 using that sampled λ. In practice, we found in prior work that DiL-piKL produces better performance.

One interpretation of DiL-piKL is that each choice of λ is an agent type, where agent types with high λ choose policies closer to $\boldsymbol{\tau}$ while agent types with low λ choose policies that are more "optimal" and less constrained to a common-knowledge anchor policy. A-priori, each player is chosen at random from this population of agent types, and the distribution represents the common-knowledge uncertainty about which of the agent types player i may be. Sampling the agent's own lambda from a probability distribution over λ values during the equilibrium computation models the fact the other players are unaware of the agent's true type.

E.2 Dialogue-conditional planning at test time

The leading approach for computing policies in dialogue-free Diplomacy applies DiL-piKL (§E.1.3), optimizing policies based on a reinforcement-learned value function with a KL regularization term towards a human imitation anchor policy (*17*). We followed this approach with important adaptations for incorporating dialogue.

E.2.1 Training a dialogue-conditional anchor policy

Similar to the dialogue-conditional action prediction model used to annotate intents for the dialogue training data described in §D.2, we fine-tuned a pre-trained Transformer seq2seq model on WEBDIPLOMACY data to predict a rollout of end-of-turn action sequences, conditional on dialogue history, draw state, order history, current game state, game metadata, and player rating (see §D.1 for a description of these text features). Specifically, we took as our base model a BART-large (*24*) model that was fine-tuned for the dialogue prediction task, in the same manner as described in §D.1.1. Starting with a dialogue model in this way induces a better prior on the Diplomacy dialogue information compared to starting from the pre-trained BART model directly. The target sequences are string representations of end-of-turn action sequences for a player, as shown in §D.2. Note, however, that in this case, we trained the model to predict action sequences for all players from the point of view of a particular player $\mathcal{A}$, rather than only training the model to predict $\mathcal{A}$'s action from $\mathcal{A}$'s point of view. This distinction is important, because $\mathcal{A}$ is only privy to the conversations it conducted directly with other players, and cannot see conversations that these players had with each other. For example, in the game turn 'Winter 1901 Adjustments', the target sequence for predicting Austria's actions from England's point of view might look like:

```
A BUD B
S1902M
A BUD GAL; A GAL SIL; A RUM S A BUD GAL; F TRI H
```

representing Austria's action sequence for the current turn as well as the future turn 'Spring 1902 Movement'. The model was trained with the same training hyperparameters as the intent model described in §D.2.

E.2.2 Sampling plausible actions for planning

At test time, we used the distribution of actions predicted by the above dialogue-conditional action prediction model as our anchor policy for applying DiL-piKL, along with a value function learned through reinforcement learning described in §E.4.

To reduce the large action space to a small enough number to apply search, we sampled 30 actions from the anchor policy for each player. While the anchor policy was the only model in this step aware of the dialogue, sampling actions from this model is computationally expensive and would also miss unusual but better actions discovered during reinforcement learning, so up to 1024 additional samples were taken from from the cheaper dialogue-free RL policy jointly learned alongside with the value function during reinforcement learning. The top 30 most likely of these actions according to the dialogue-free RL policy were then rescored for their likelihood under the dialogue-conditional anchor policy. Of the final 60 actions, 30 from the anchor policy and 30 from the dialogue-free RL policy, the top 35 most likely according to the anchor policy were kept. These final 35 actions per player, along with their relative probabilities, were used as a tabular approximation to the anchor policy for the remainder of the search procedure.

E.2.3 Independent pairwise policy prediction

DiL-piKL (§E.1.3) can be directly applied to Diplomacy to approximate an equilibrium across all 7 players at once, but unlike its successful application to dialogue-free Diplomacy in (*17*), for the full game of Diplomacy this approach is lacking in two respects:

1. It fails to account for correlation between the actions of other players based on dialogue that the agent cannot observe. For example, Germany may have agreed via private dialogue to support France to *either* Ruhr or Munich with France moving to precisely whichever same territory that Germany is supporting, but the agent doesn't know which.

2. It fails to account for the fact that the dialogue with each player is not common knowledge. For example, if the agent playing as Italy discusses with Austria moving to Venice, France shouldn't be able to adjust their policy based on this (secret) dialogue.

Therefore, we computed pairwise policies between the agent player i and each other player $j \neq i$. One at a time, we ran DiL-piKL between just i and j in order to obtain a predicted policy π_j for that other player.

Specifically, we ran DiL-piKL where on each iteration t, the agent i and player j respectively sampled anchor strengths λ_i^t and λ_j^t and respectively sampled actions $a_i^t \leftarrow \pi_i^t(\lambda_i^t)$ and $a_j^t \leftarrow \pi_j^t(\lambda_j^t)$ from the piKL policies $\pi_i^t(\lambda_i^t)$ and $\pi_j^t(\lambda_j^t)$ on that iteration. For the other 5 players, we applied the computationally cheap dialogue-free RL policy π_{RL} to sample a joint action conditional on the actions of i and j: $a_{-ij}^t \leftarrow \pi_{RL}(a_i^t, a_j^t, *, *, *, *, *)$. Sampling a joint action in this way allowed us to account for likely possible correlation as a result of unobserved private dialogue between those players, or between those players and j.

Equipped now with a joint action $\boldsymbol{a} = \{a_i, a_j, a_{-ij}\}$, we performed a 1-step rollout following a and apply the RL value model to the resulting state, obtaining rewards u_i^t and u_j^t for that DiL-piKL iteration. We performed a total of 256 iterations with each player, obtaining separate policies π_j for each player j.

These policies represent a best guess for the likely range of actions that each other player may play, having been regularized at each step to remain a close match to the dialogue-conditional human imitation anchor policy, while through piKL downweighting low-value and/or exploitable actions. The π_j are the final policies from which we selected actions for message recipients to supply as intents (§D.2).

E.2.4 Choosing the agent's own action and/or intent

In order to compute the agent's own action, our goal was to play a best response to a joint combination of all π_j for all the players j other than the agent i. However, since each π_j was computed separately, we lacked the information about how these policies correlate with one another.

Therefore, we merged all policies back into a single correlated joint policy via self-normalized importance sampling by reapplying the correlations learned during RL. Specifically, we sampled 1000 joint actions $\boldsymbol{a}_{-i}^k,\ k \in \{1, \ldots, 1000\}$ from the cartesian product of all these policies, $\forall j \neq i :\ a_j^k \sim \pi_j(\cdot)$.

Then for each $\boldsymbol{a}_{-i}^k$, we weighted it by the likelihood ratio:

$$w(\boldsymbol{a}_{-i}^k) = \pi_{RL}\left(\boldsymbol{a}_{-i}^k\right) / \left(\prod_{j \neq i} \pi_{RL}(a_j^k)\right).$$

That is, the ratio between likelihood of $\boldsymbol{a}_{-i}^k$ under the (correlated) joint RL policy, and the product of the likelihood of a_{-i}^k under the independent marginal RL policies.

The reweighted average is the final estimate of the joint policy of players other than the agent:

$$\pi_{-i}(\boldsymbol{a}_{-i}) \propto \sum_k w(\boldsymbol{a}_{-i}^k) I(\boldsymbol{a}_{-i}^k = \boldsymbol{a}_{-i})$$

Lastly, we chose the action a_i for the agent that best responded to this predicted joint policy π_{-i} of the other players, but again regularized towards the anchor policy τ. Specifically, we chose the action $\operatorname{argmax}_{a_i} u_i(a_i, \boldsymbol{\pi}_{-i}) + \lambda \log \tau_i(a_i)$ where u_i is the RL value function, $\tau_i(a_i)$ is the probability of the action under the dialogue-conditional anchor imitation policy and $\lambda = 0.003$. Although λ is small, it resulted in the agent preferring consistency with the dialogue if multiple actions had similar predicted values.

E.2.5 Distribution of the anchor strength λ in DiL-piKL

For all applications of DiL-piKL in CICERO, we used a uniform distribution over six possible λ values log-uniformly spaced, where the smallest λ was 0.001 and the largest λ was 0.3. In particular, by using a wide a distribution of anchor strengths during pairwise prediction, we aimed to make CICERO favor policies that would be robust to a wide range of opponents, from those that would closely adhere to the BC policy prediction and/or the likely actions implied by their dialogue with CICERO, to those that would deviate more heavily.

However, we additionally multiplied all λ values by a factor of $(1/c)\sqrt{\text{Var}_i(s) + \epsilon^2}$, where $\text{Var}_i(s)$ is the empirical variance of player i's value estimate over 100 rollouts of the raw model's policy from the current state s up to the next negotiation/movement turn, $c = 0.05$ normalizes for the typical standard deviation for value estimates at the start of the game, and $\epsilon = 0.01$. Recall that the anchor strength λ is the additional expected reward from a better action at which piKL predicts a player to have an e times increased odds to choose that better action. The motivation for this scaling was to address an informal observation that in situations where the overall variance of values of all actions is larger or more uncertain, human players might be less sensitive to same absolute value difference in adjusting their policy, so piKL should adjust less. Or stated another way, when the variance of state values is high, the anchor strength may also need to be correspondingly higher to achieve a "similar" overall level of regularization towards the BC policy.

Lastly, we also multiplied all λ values by 10 for the first turn of the game, so that CICERO modeled other players as playing, and itself played, a range of openings which more closely resembles the variety of openings that humans typically play.

E.3 Dialogue-free model inputs and architecture

This section describes the architecture used for human behavioral cloning and for reinforcement learning in the dialogue-free part of the agent architecture.

Whereas we rely heavily on dialogue-conditional language generation and dialogue-conditional action proposal models during test-time strategic planning (§D.2, §E.2), we do not address the challenges of running reinforcement learning with natural-language dialogue "in the loop" at training time. As a result, the reinforcement-learned value function and policy (§E.4) that drive the various steps in strategic planning in §E.2 use a model that is similar to language-free architectures used for much prior published work in dialogue-free Diplomacy (*7, 16, 26, 53*).

This dialogue-free part of CICERO's architecture takes as input the board state and recent action history, plus a set of zero or more given orders (possibly including orders for more than one player) to condition on as orders that will be played this turn. Via an autoregressive policy head it predicts the likelihood of all *other* orders by all players this turn conditional on the given orders. Either jointly on the same model or attached to a separately trained model with the same architecture, a value head predicts the expected value of the final score for all players. In this way, the policy head can model the joint distribution of the likely actions of human players in a game when trained via BC, or the joint distribution of human-like policies after those policies are improved via RL, while the value head is trained to approximate the expected continuation value of the game under that joint policy. Conditioning on given orders in the input enables us to also query conditional slices of that distribution rather than only sampling from the distribution as a whole.

Although this model does not directly see the dialogue in the game, by predicting the full joint distribution of all player's actions, it can model likely correlations between different players' actions due to those players coordinating through private unseen dialogue. As described in §E.4, we leverage this in RL to be able to regularize our RL policy to remain close to the human joint policy inclusive of correlated actions coordinated through private dialogue, despite not receiving dialogue as an input nor explicitly modeling dialogue during RL.

Similar to prior work in dialogue-free Diplomacy (*7, 16, 26, 53, 54*), we encode the board state as a tensor of the 38 feature channels described in Table S7 for each of the 81 possible locations on the board (75 regions plus 6 special coasts).

We encode both the current board state and the previous board state in this way. We also encode recent order history the same way as in (*53*), where previous turn orders are associated with the board location of the unit performing that order and are encoded as a vector of length 202 indicating the order properties. We borrow the same embedding as well for the current turn orders being conditioned on, with an all-zero embedding for orders not given.

All of these components are concatenated channel-wise for a total of 480 channels, and passed through a linear layer with 224 output channels, producing a 81 (location) x 224 (channel) input tensor encoding all per-location information about board state.

Similarly, each component of information about the board state that is per-player (Table S8) or global (Table S9) is linearly processed and summed in the same way, producing a final 7 (player) x 224 (channel) tensor and a 1 (global) x 224 (channel) tensor.

All tensors are concatenated to a single (81+7+1) x 224 tensor, summed with a randomly-initialized learnable positional bias, and fed to a standard transformer encoder architecture. The encoder uses 10 encoder blocks, where each block consists of a self-attention layer with 224 channels, 8 softmax dot-product self-attention heads, followed by a 224x224 channel-wise-fully-connected transform with GeLU activations and skip connection from the previous block.

The final output policy is predicted from the encoder's output autoregressively by an LSTM decoder head that exactly matches that of (*53*), except with the only difference being that the LSTM decodes over all locations corresponding to any player's possible orders, rather than only the locations for which one player can issue orders, so as to decode and predict the joint and possibly correlated actions of all players, rather than only a player's individual action.

Separately, the game value is predicted by a value head that takes the transformer encoder output and applies a softmax attention layer, a linear layer with 224 channels, GeLU activation, a linear layer with 7 channels, and a softmax, giving the predicted expected final score for all 7 players.

See Figure S7 for an illustration of the architecture.

Pre-RL behavioral cloning We used the WEBDIPLOMACY dataset to train a baseline behavioral cloning model using this architecture before improving that model further through RL (§E.4). Our training was similar to that of (*17*), except that the model was trained to predict joint actions of one, two, or all players, rather than only one player actions. During BC training, we included games both with and without dialogue, and at inference time and during RL, we set the model to condition on the game being regular Diplomacy rather than dialogue-free Diplomacy (as per the appropriate input in Table S9). See Table S10 for a summary of the hyper parameters used. To train a pre-RL value function, we also trained a separate model from rollouts of the behavioral cloning policy using the improved value modeling method from (*26*).

E.4 Correlation-aware human-regularized reinforcement learning

Strategic planning requires accurate value estimates in diverse situations, as well as the ability to surface the best actions. Unlike 2p0s games where there is a uniquely defined minimax value for each state s, to do well in multiplayer games among human players, we need a value estimate $V(s)$ assuming a human distribution of continuation policies. However, value models trained solely on human data suffer from high variance due to limited data. There are only about 100,000 final game outcomes in the WEBDIPLOMACY dataset, and the final outcome is an extremely noisy estimator of the value of an intermediate game state.

Self-play reinforcement learning, in which an agent learns by playing repeated games against earlier copies of itself, has shown remarkable success in learning high-quality policy and value functions even when training from scratch without human data in a variety of adversarial games including Go (*90*), chess (*27*), poker (*4, 10*), and Dota 2 (*32*). However, we found that naïve self-play RL from scratch performed poorly in a dialogue-free variant of Diplomacy when playing with humans (*17*). This is because in non-2p0s games there may be multiple incompatible equilibria, and naïve self-play RL may converge to state values that differ greatly from the correct value for human play. Moreover, in a game like Diplomacy where private communication is possible, one also has to model how players might coordinate and correlate their actions through private messages, including messages the agent might not observe.

We therefore developed Correlated and Shared (CoShar) piKL, which regularizes toward a joint, correlated anchor policy τ shared by all players rather than toward per-player policies. This allows both keeping a policy from drifting too far from human conventions during optimization as well as allowing for cross-player correlations to be modeled in a joint policy. We then shaped our reinforcement learning around CoShar piKL.

CoShar piKL operates by iteratively improving a policy π^t over successive iterations t based on adjusting a joint anchor policy τ using the average expected value of the policies on past iterations:

$$Q_i^{t-1}(a_i) = E_{a\sim\pi^{t-1}(a|a_i)}V_i(a) \tag{S6}$$

$$\pi^{\Delta t}(a) \propto \tau(a)\exp\left(\sum_i Q_i^{t-1}(a_i)/\lambda\right) \tag{S7}$$

$$\pi^t = \left(\frac{t-1}{t}\right)\pi^{t-1} + \left(\frac{1}{t}\right)\pi^{\Delta t} \tag{S8}$$

where expected values are given by a value function V such that $V_i(a)$ is the value to player i of the game state resulting from playing joint action a, and the anchor strength λ limits the degree to which the policy can deviate from τ.

When the individual player policies in τ are uncorrelated, CoShar piKL is identical to piKL.

Unfortunately running CoShar piKL is not computationally feasible in Diplomacy due to the combinatorial action space. For example, if all 7 players each have 35 actions to choose from, the joint action space is of size $35^7 > 10^{10}$, too large to be cheaply enumerated. Sampling from π^t is also nontrivial, due to the scaling by Q.

Therefore, we developed a simplified version of the algorithm, Correlated Best Response, that computes a KL-regularized best response in an efficient way. We then combined it with self-play to iteratively improve our estimates of the joint policy.

E.4.1 Correlated Best Response

Given a current policy π_{θ_π} parametrized by a neural network θ_π, Correlated Best Response (Cor-BR) aims to perform only a single-step of CoShar piKL at a time to improve that policy. Therefore, to derive the update rule from Equation S6, we remove the iteration index t as we are updating the policies only one step, and draw conditional samples from π_{θ_π} to estimate EVs.

$$Q_i(a_i) \quad = \quad E_{a \sim \pi_{\theta_\pi}(a|a_i)} V_i(a), \tag{S9}$$

$$p(a) \quad \propto \quad \tau(a) \exp\left(\sum_i Q_i(a_i)/\lambda\right). \tag{S10}$$

Note, that the first step (S9) in this formulation is significantly more efficient than in Equation S6 because $\pi_{\theta_\pi}$is modeled by a neural network that allows sampling conditional joint actions in linear time, whereas π^{t-1} in CoShar piKL may not be easy to sample. Here p is analogous to $\pi^{\Delta t}$ from CoShar piKL, it is the policy which we wish to take a small step towards in order to improve our current policy π_{θ_π}.

Next, we attempt to sample from p. To sample from p we employ a self-normalized importance sampling trick similar to §E.2.4, as follows.

Let q be a product of the independent piKL policies for each player, that is:

$$q(a) \quad \propto \quad \left(\prod_i \tau(a_i)\right) \exp\left(\sum_i Q_i(a_i)/\lambda\right). \tag{S11}$$

The distribution q would match p if τ factorized as a product of independent marginal policies for each player, i.e if there were no private communication by which players could correlate their actions. But since there are correlations, we can sample from q and then sample from these samples via importance sampling, using the ratio of probabilities $p(a)/q(a)$ as the weight of each sample. More formally, the full procedure of sampling N_p many joint actions $a^1, \ldots, a^{N_p}$ from Correlated Best Response policies goes as follows:

$$\begin{aligned}
\hat{Q}_i(a_i) \quad &:= \quad \mathbb{E}_{a \sim \pi_{\theta_\pi}(\cdot|a_i)} V_i(a) \\
\hat{q}(a) \quad &:= \quad (\prod_i \tau(a_i)) \exp(\sum_i \hat{Q}_i(a_i)/\lambda) \\
\hat{p}(a) \quad &:= \quad \tau(a) \exp(\sum_i \hat{Q}_i(a_i)/\lambda) \\
\hat{a}^1, \ldots, \hat{a}^{N_q} \quad &\sim \quad \hat{q}(\cdot) \\
\hat{\mathfrak{p}}(a) \quad &:= \quad \hat{p}(a)/\hat{q}(a) I(a \in \{\hat{a}^1, \ldots, \hat{a}^{N_q}\}) \\
a^1, \ldots, a^{N_p} \quad &\sim \quad \hat{\mathfrak{p}}(\cdot)
\end{aligned}$$

Note that here $\hat{p}$, $\hat{q}$, and $\hat{\mathfrak{p}}(a)$ are unnormalized probability distributions, but computation of the normalization constants for the latter two is tractable. We can sample from $\hat{q}$ as we can sample actions independently. We can sample from $\hat{\mathfrak{p}}(a)$ as only a small set of actions we sampled from $\hat{q}$ have non-zero probability. As $N_q \to \infty$ the probability distribution of final samples converges to p.

Lastly, we can approximately update π_{θ_π} towards p by a small step by taking a gradient step training θ_π to minimize cross entropy loss in predicting the distribution of final samples.

Effectively, we are doing the same thing as piKL, approximately computing a regularized equilibrium where all players attempt to optimize their own utility subject to their policy remaining close to the human imitation anchor policy, so that their conventions do not drift far from human conventions. With two differences:

- As with CoShar piKL, rather than regularizing the policies of all players independently towards their own marginal action distribution, we regularize the joint policy of all players towards the joint imitation anchor policy, so as to preserve the correlation structure between different players' actions that results from their ability to privately communicate.
- But rather than running the equilibrium computation to completion at every point of the training trajectory, we only run a single gradient step of it at every point of the training trajectory, amortizing the optimization over the entire RL training loop.

E.4.2 Human correlation aware self-play

Our resulting self-play algorithm, RL-Cor-BR, closely followed Deep Nash Value Iteration (DNVI) (*16*), as well as our prior work in RL in dialogue-free Diplomacy (*17*), except replacing the planning algorithm from regret matching or DiL-piKL, respectively in those works, with Cor-BR. Our implementation similarly operated several data generation workers and training workers that run in parallel.

The data generation workers played games using the current estimates of the value and policy proposal networks. On each turn of a game we computed a Corr-BR joint policy as described in the prior section using a fixed λ of 0.03 with 100 samples for approximating p, and computed the estimated state value under this policy based on a 1-step rollout. We added the policy and value to a circular replay buffer shared among all workers and sampled an action from the policy to play to continue the game.

Simultaneously, the training workers continuously read batches of data from the replay buffer, synchronously applied a gradient update to the value and policy proposal networks, and periodically broadcasted the updated models to the data generation workers.

For the value function we used the same loss function for RL as for imitation learning:

$$\text{ValueLoss}(\theta_v) = \frac{1}{2}\left(V(s;\theta_v) - \sum_{a'} \sigma(a') V\left(f(s,a');\hat{\theta}_v\right)\right)^2$$

The resulting value function produces high-quality state-value estimates taking into account how players' ability to privately coordinate can affect the likely future outcomes under human-like conventions of play, despite not directly observing the dialogue.

To train the policy, since for the CICERO planning in §E.2 we needed the RL policy to be able to predict the joint policy of players conditioned on anywhere from zero to two players' actions, we constructed policy training targets accordingly.

To do so we first sampled a random joint action a from the policy. We then sampled a vector of conditioning actions c that randomly either was empty, or contained one random player's action, or two random players' actions. Lastly, we then used a standard cross entropy loss:

$$\text{PolicyLoss}(\theta_\pi) = -\log \pi_i(s, a; \theta_\pi | c)$$

As described in §E.2, the resulting RL policy was used to augment samples from a dialogue-conditional action prediction model, as well as to provide a fast estimate of other players policies during pairwise DiL-piKL search.

E.4.3 Evaluation of RL policy and value function

The goal of coordination-aware training is to be able to able to understand and play well with humans. However, running many tests against real humans beyond the ones we have run for our main experiments is logistically difficult and impractical. As a proxy we performed population and head-to-head experiments to measure the strength of agents using various policy and value functions as well as some experiments to probe and verify that the RL policy and value networks nontrivially model a correlated multi-agent human-like policy rather than one that could simply be factored as the product of per-agent independent policies.

To test the value function, we compared the effects of using RL-DiL-piKL (from (*17*)) which optimizes independent policies for each player, versus the RL-Cor-BR that we actually used in CICERO that captures cross-player correlations in actions. We also tested “independent improved BC” training of a value model described by (*26*), and an analogous version of the same that uses rollouts from the joint BC policy rather than the independent BC policies for each player.

We find in Table S11 that the RL-Cor-BR value function performed significantly better than all other tested methods against coordinating agents, whereas the RL-DiL-piKL hardly improved over BC methods, suggesting that RL-Cor-BR's learned state values do meaningfully capture the effects of future multiplayer coordination on the current value of a position.

Notably, RL-Cor-BR performed worse than RL-DiL-piKL by average score in a population of independently-acting search-based agents, which is also consistent with expectations — in a population of agents that do not in fact correlate their actions, a value function trained to expect correlation underperforms one that correctly assumes independence. Still, both algorithms outperform BC methods.

To evaluate RL policies, we evaluated how well the agent acting directly according to that raw policy could play against 6 coordinating agents as well as how the 6 copies of that agent sampling correlated actions from that policy could coordinate against a single search agent (Table S12). We compared Cicero's dialogue-free RL policy versus dialogue-free imitation policies trained to predict individual player actions independently per player, or to predict joint actions of all players.

As expected, in the 1 row agent vs 6 coordinating BC agent setting, the independent BC policy and joint BC policy performed identically, since sampling independently the single agent's policy versus sampling joint actions and marginalizing down to the single agent's policy produces theoretically the same policy. The RL policy from CICERO outperformed both.

By contrast, in the 6 coordinating row agents vs 1 BC search agent setting, the joint BC policy outperformed the independent BC policy because the joint BC policy was able to sample correlated actions among the 6 players it controls modeling the way those players would likely have coordinated due to private dialogue (despite not explicitly simulating such dialogue). CICERO's RL policy yet further outperformed that, indicating the value of RL over just plain BC.

F List of models used in CICERO

We list and briefly describe the deep-learning models used in CICERO and/or featured in this work. For each model, we also briefly summarize its inputs and training, including in what way, if any, it implicitly or explicitly accounts for dialogue.

- Intent-controlled dialogue model (main article section Intent-controlled dialogue, SM D.1) - Responsible for all dialogue generation in CICERO. Trained to imitate messages in human diplomacy games given the board state and history, the dialogue history, and the intent for the target message. At test time, the intent was supplied from strategic planning instead.

- Intent annotation model (main article section Annotating training messages with intents, SM D.2.3) - Used to annotate the human dataset messages with intents, for training the intent-controlled dialogue model and imitation intent model. Not used in CICERO at test time. Trained to predict the actions in human Diplomacy games given the board state and history, and the dialogue history. At annotation time, the dialogue history included messages up through the message being annotated, with additional agreement messages injected.

- Imitation intent model (SM D.2.4) - Used to determine actions that humans would be likely to talk about in order to select among actions in a recipient's predicted policy to use as an intent for generating a message to the recipient. Trained on human Diplomacy games to predict the annotated intent for the next message, given the board state and history and dialogue history prior to that next message.

- Message filtering ensemble models (main article section Message filtering, SM D.3.1) - Used to filter CICERO's generated dialogue for inconsistencies and nonsensical messages. Includes 16 classifiers, each trained to discriminate between real human messages from human Diplomacy games and messages with different deliberate corruptions applied, or generated by different models, and provided different inputs (different amounts of board state and history, or dialogue history, etc).

- Weak/denoising models (SM D.3.1) - Used to generate counterfactual and/or corrupted messages for training message filtering ensemble models. Not used in CICERO at test time. Variously trained to predict or generate dialogue given limited information, so as to generate messages that contain errors to be discriminated from accurate human messages.

- Message scheduler model (SM D.4.1) - Used along with other heuristics to determine whether to send any (further) messages to a player during a turn. Trained on human Diplomacy games to classify whether and when to send the next message to a particular recipient, given the board state and history and dialogue history (but not the contents of the message to be sent). At test time, used only to determine whether to send a message to a given player, not when.

- Dialogue-conditional BC model (main article section Strategic reasoning, SM E.2.1 - Supplied the anchor policy by which CICERO's strategic reasoning stays compatible with human conventions, allows CICERO's actions to be highly flexible and adaptive to the dialogue it receives from other players. Trained on human diplomacy games to imitate/predict the joint actions of all 7 players, given the board state and history and the dialogue history. At test time, used as the anchor policy for piKL, as well as to initially sample actions to be considered by piKL.

- Dialogue-free BC policy model (SM E.3) - Supplied the anchor policy used during RL to ensure the RL policy stayed compatible with human conventions. Not used in CICERO at test time. Trained on human diplomacy games to imitate/predict the joint actions of all 7 players, given the board state and history but without observing the dialogue. Still implicitly captures some effects of dialogue on likely actions, such as predicting correlations between the actions of players who are likely to coordinate through private dialogue.

- Dialogue-free Improved BC value model (SM E.3) - Used only to initialize the dialogue-free RL model, and in an ablation. Not used in CICERO at test time. Trained from the dialogue-free BC policy following the improved value modeling method from (*26*) to predict the expected final score of players, given the board state and history but without observing the dialogue.

- Dialogue-free RL policy model (main article section Self-play reinforcement learning for improved value estimation, SM E.4) - During pairwise piKL in CICERO, used to sample actions for the other 5 players conditional on the actions of the pair. Also used to sample initial actions to be considered by considered by piKL. Trained via RL to predict the piKL policy anchored to the dialogue-free BC policy, given the board state and history but without observing dialogue. Intended to be a better policy than BC that still accurately models human play, including cross-player correlations due to the unobserved dialogue.

- Dialogue-free RL value model (main article section Self-play reinforcement learning for improved value estimation, SM E.4) - Used for all value estimates in CICERO during strategic reasoning. Trained via RL jointly with the RL policy, to predict the expected continuation value of piKL anchoring to the dialogue-free BC policy, given the board state and history but without observing dialogue. Still implicitly captures some effects of possible future dialogue on state values due to being the continuation value of a joint policy that, e.g., implicitly models players' likely future coordination.

G Performance optimizations for blitz Diplomacy

Playing in blitz games requires the agent to respond to messages quickly and send a lot of messages in a short time frame. We made the following performance optimizations in order to keep the agent responsive and able to respond to most messages within 15-30 seconds.

Hardware setup and parallel agents CICERO can sample policies, perform search, and generate candidate messages in parallel on multiple GPUs on a single machine. In blitz games, we ran on a machine with 80 Intel(R) Xeon(R) CPU cores and 8 NVIDIA(R) V100 GPUs. We used data-parallelism to speed up many operations such as message sampling, order sampling, state value evaluation, etc. Message generation time was 10-30 seconds, depending on the length of dialogue history and the number of units controlled by the agent and recipient. In order to further improve responsiveness, we designated a separate machine to control the dialogue with each recipient, and one machine that updated orders in response to the dialogue. These "recipient sub-agents" operated wholly independently in response to game events.

Interruption Typically, any game event (incoming messages or new turn) would re-start message generation. In blitz games, an agent is often interrupted by incoming messages faster than it can generate messages, which would prevent it from ever speaking. Therefore, we set each recipient sub-agent to only be interrupted by incoming messages from the recipient it was speaking to; messages would therefore be slightly stale with respect to dialogue with other powers.

Caching and incremental planning We cached and reused intents for multiple outgoing messages when not receiving new incoming messages. In other cases when we already sent a message on a turn, we incrementally updated the tabular anchor policy by sampling 10 additional actions from the dialogue conditional model for the agent and each power who has spoken since the last message, rescoring everything with the dialogue-conditional model.

Intents for future turns Message intents consist of actions for the agent and recipient for both the current turn and future turns through the next movement turn. To reduce compute time, we only incorporated planning when generating intents for the current turn, using the imitation intent model alone (conditioned on current turn intents) to generate intents for future turns.

H Results of Anonymous Human Games

Table S13 lists the average score and number of games played with CICERO for every player who participated in games with CICERO. Figure S9 visualizes CICERO's average score compared to the cumulative distribution of players, restricting to those who played at least 1, 2, 3, and 5 games respectively.

Table S14 compares cross-player support statistics for CICERO and other ways of generating orders in the league games played on webDiplomacy.net. We compared statistics for CICERO's orders with CICERO orders at the start of the phase, before any dialogue is exchanged, in order to measure the effect of coordination through dialogue. As a more extreme example, we also considered the orders that a state-of-the-art agent for no-press Diplomacy, which ignores all dialogue, would play. We find that CICERO has more effective and coordinated cross-player supports than either of these variants. However, CICERO is still less coordinated than pairs of human players.

I Simulated Diplomacy games between bot variants

We evaluated a variety of ablations and alternative Diplomacy agents in head-to-head matchups with one agent of type A and six agents of type B. For each matchup, we ran 40 games with agent A playing each of the seven powers, for a total of 280 games. Results are shown in Table S15. For agent B we used either an "imitation agent" or CICERO. The imitation agent used a dialogue-conditional imitation model to select actions $P(a|x)$ and intents $P(z|x)$. In other words, this agent imitated the human distribution of play (conditioned on a high player rating). We performed fewer matchups with CICERO as agent B because of the computational cost of running 6 CICERO agents.

The results of these experiments demonstrate the importance of some aspects of CICERO, but also highlight the limitations of evaluating a dialogue agent through self-play. Notably, the imitation agents clearly capture the importance of communication for cooperative behavior: silent agents, even those with excellent strategic play, perform quite poorly against them. However, while the performance against 6 imitation agents seems to be almost perfectly correlated with the number of messages the agent sends, the effect of the content of these messages is not observed (Figure S10). Indeed, removing intents or filters which we have shown to decrease nonsense leaves the performance against this population the same or better, by allowing more messages to be sent. This reflects the fact that imitation agents learn an artificial correlation in the data between a large number of friendly messages with a player and friendly behavior towards that player, allowing them to be easily exploited. On the other hand, talking more to a CICERO agent has a negligible effect on outcomes.

The benefits of strategic planning are also evident in table S15. CICERO achieved an average score of 32% against six imitation agents, while an imitation agent achieved an average score of 4.5% against six CICERO agents. We also ablated the RL models and instead used a supervised-learned value function trained only on human games for planning, which reduced average score from 32.0% to 25.8%. This reflects the benefit of RL over supervised training of a value function.

	Info	Text
Input	(Truncated) previous turn dialogue history	... 16 TURKEY -> AUSTRIA: my moves are AEG - ION, and BUL s SER - RUM 1848 AUSTRIA -> TURKEY: Awesome! Thanks. F1902M
	Current turn dialogue history	0 TURKEY -> AUSTRIA: wait, if you move into RUM, how are you going to move TRI to SER? 574 TURKEY -> AUSTRIA: i'll do it anyway, so please confirm
	Draw state	DRAWS:
	Order history	S1902M ENGLAND: A NWY H; F LON ENG; F NTH S F LON ENG; F NWG S A NWY FRANCE: A BEL S A MAR BUR; A MAR BUR; A PAR PIC; A SPA GAS; F BRE ENG; F POR MAO ITALY: A TUN H; A VEN H; F ION EAS; F NAP ION GERMANY: A BER KIE; A HOL H; A MUN H; F DEN S F KIE HEL; F KIE HEL AUSTRIA: A BUD SER; A SER BUL; A TRI S A BUD SER; A VIE BUD; F GRE S A SER BUL TURKEY: A BUL H; A CON S A BUL; F ANK BLA; F SMY AEG RUSSIA: A FIN S F STP/NC NWY; A RUM S A WAR GAL; A WAR GAL; F SEV BLA; F STP/NC NWY; F SWE S F STP/NC NWY S1902R
	Game state	units: AUSTRIA: A BUD, A SER, A TRI, A VIE, F GRE; ENGLAND: F ENG, F NTH, F NWG; FRANCE: A BEL, A BUR, A GAS, A PIC, F BRE, F MAO; GERMANY: A HOL, A KIE, A MUN, F DEN, F HEL; ITALY: A TUN, A VEN, F EAS, F ION; RUSSIA: A FIN, A GAL, A RUM, F NWY, F SEV, F SWE; TURKEY: A BUL, A CON, F AEG, F ANK centers: AUSTRIA: BUD, GRE, SER, TRI, VIE; ENGLAND: EDI, LON, LVP, NWY; FRANCE: BEL, BRE, MAR, PAR, POR, SPA; GERMANY: BER, DEN, HOL, KIE, MUN; ITALY: NAP, ROM, TUN, VEN; RUSSIA: MOS, RUM, SEV, STP, SWE, WAR; TURKEY: ANK, BUL, CON, SMY
	Planned orders (recipient)	AUSTRIA: A BUD S A SER RUM; A SER RUM; A TRI SER; A VIE GAL; F GRE S F AEG ION F1902R W1902A A TRI B S1903M A BUD S A RUM; A RUM S A SER; A SER S A RUM; A TRI S A VIE TYR; A VIE TYR; F GRE S F AEG ION
	Planned orders (sender)	TURKEY: A BUL S A SER RUM; A CON SMY; F AEG ION; F ANK BLA F1902R W1902A S1903M A BUL S A RUM; A SMY H; F AEG ION; F ANK BLA
	Time since last message	6225
	Player and game metadata	F1902M TURKEY 5 ANON 1440min SOS PUBLIC HASDRAWS:
Target		AUSTRIA: i want you to support me into the ION, i can't have italy in the AEG

Table S1: **Hypothetical dialogue training example** All dialogue shown is model-generated. *Info* denotes the game-related information that is being represented in the particular part of the text sequence.

Method	% of predictions containing labeled orders
Base model	77
+ Initialized from dialogue model	87
+ Injected agreement messages	93
+ Restriction to truthful subset	97

Table S2: **Intent annotating test set results** We show the results on a small (194 example) test set evaluating how closely the predicted intents correspond to the dialogue messages. The test set consists of dialogue messages hand-labeled with orders reflected in the content of the dialogue message. Adding each successive method used by Cicero substantially improves the correspondence between intent and dialogue.

Example of contradiction – Cicero is **Austria**
...
Austria: Also, are you able to move Ven-pie and Apu-ven?
Italy: Yeah that's what I'm planning
...
Austria: I'm not a fan of the move to Venice. Are you planning to hold in Venice?
Italy: You suggested I move to Venice!
Italy: Do you want support to Aegean this turn?
Italy: And yes Venice will hold
Austria: In that case, we're all good! And yes please!

Table S3: **Unsuccessful dialogue example** Cicero contradicts a message asking Italy (an author of this paper) to move to Venice. Although our suite of filters aims to detect mistakes of this nature, it is not perfect.

(Model #)	Context	Corruption type
Context length: 512 tokens		
(1)	messages, state	location (entity)
(2)	messages, state	power (entity)
(3)	dialogue history, state	symbol (entity)
(4)	messages, state	weak
Context length: 2048 tokens		
(5)	orders (1 M-phase), state, intents	denoising (seed 1)
(6)	messages, orders (2 M-phases), state, intents	denoising (seed 1)
(7)	messages, orders (2 M-phases), state (2 M-phases), intents	denoising (seeds 1, 2)
(8)	messages (no speaker), orders (2 M-phases), state (2 M-phases), intents	denoising (seed 1)
(9)	messages, orders (4 M-phases), state, intents	denoising (seed 3)
(10)	messages (no speaker, bilateral), orders (2 M-phases), state (2 M-phases), intents	denoising (seed 1)
(11)	messages, orders (2 M-phases), state (2 M-phases), intents	weak justifications
(12)	messages, orders, state, intents	non-sequiturs
(13)	messages, orders (2 M-phases), state, intents	denoising (seeds 1,2,3)
(14)	messages, orders (2 M-phases), state, intents	denoising (cardinals)
(15)	messages, orders (2 M-phases), state (2 M-phases), intents	denoising (seeds 1, 2)
(16)	messages, orders (2 M-phases), state, intents	denoising (negations)

Table S4: **List of nonsense classifiers in the ensemble.** We detail the list of all 16 classifiers used in the ensemble used for detecting nonsense (§D.3). We describe the different input contexts that each of these models were trained with, as well as the 'type' of counterfactuals that were used as negatives during training.

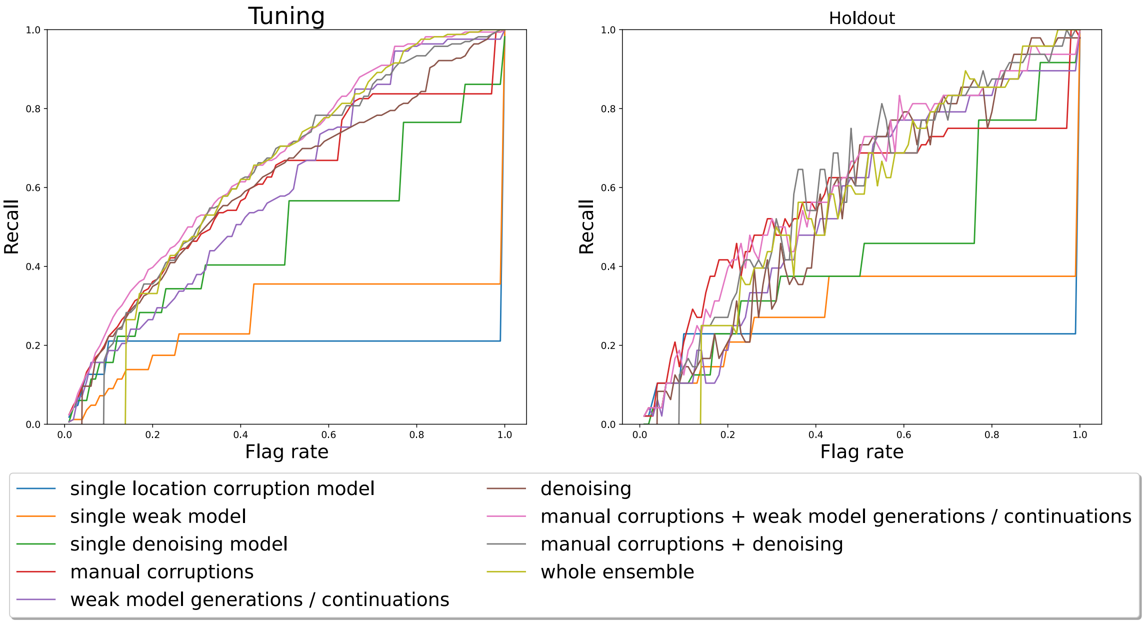

Figure S2: **Trade-off between recall and proportion of messages filtered for nonsense ensemble on annotated test dataset** These graphs illustrate the trade-off between the rate of filtering messages (flag rate) and proportion of nonsense filtered (recall), as measured on our annotated test dataset. Ensembles tend to significantly outperform individual classifiers. While simpler combinations of classifiers than the entire ensemble reach similar levels of recall, there are additional factors to consider when choosing classifiers, such as their effects on message diversity.

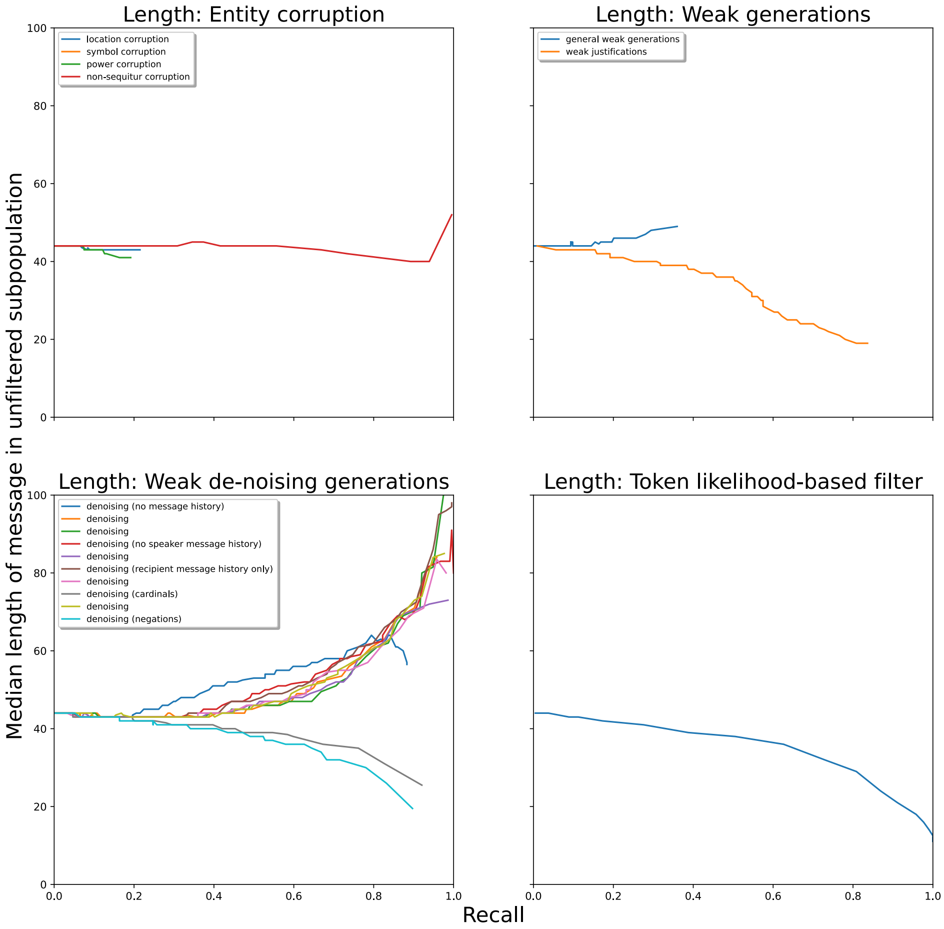

Figure S3: **Effect of nonsense filtering on message diversity (message length)** We compute median lengths of the remaining messages in our annotated nonsense dataset after filtering by each nonsense classifier at different thresholds, plotting the length against the recall (proportion of nonsense filtered) at that threshold. Length is used as a proxy to understand how nonsense classifiers affect diversity of dialogue. Most weak de-noising-based classifiers do not show a negative effect on message length. Only two do, along with the token-likelihood-based classifier and one of the weak generations classifiers.

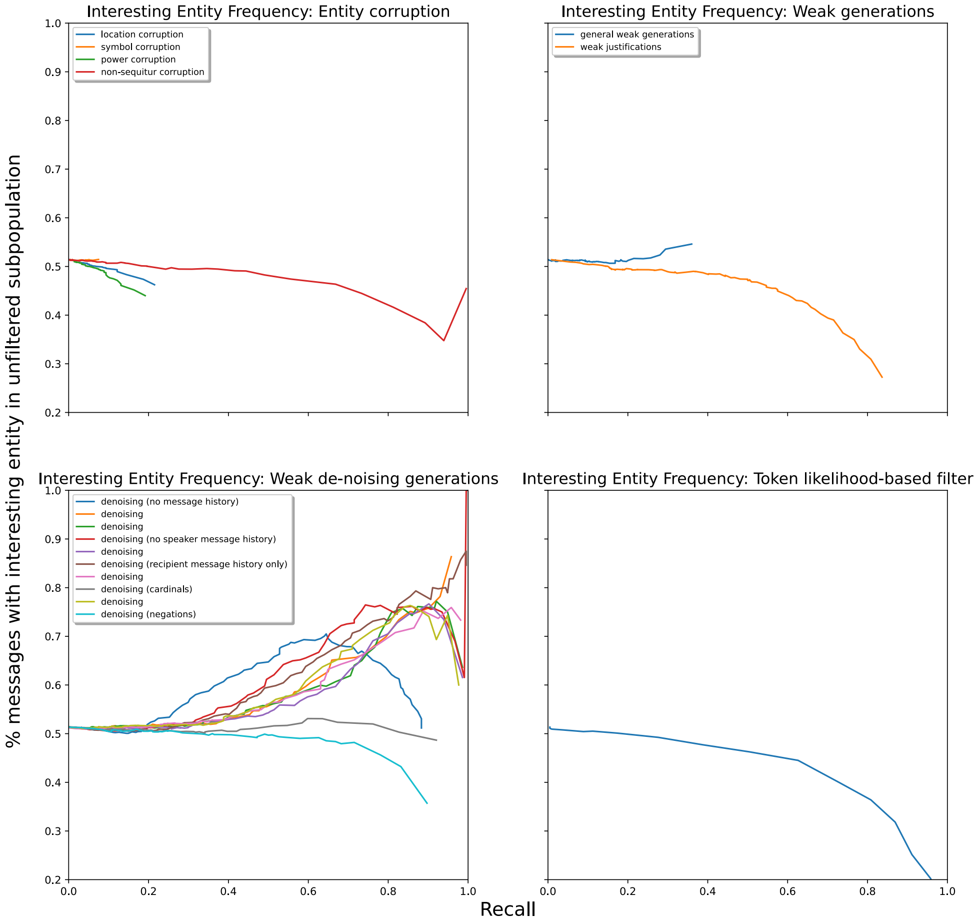

Figure S4: **Effect of nonsense filtering on message diversity (entity mentions)** We compute proportions of "interesting entities" in remaining messages after filtering messages by each nonsense classifier at different thresholds, plotting the proportion against the recall (proportion of nonsense filtered) at that threshold. This is used as another proxy to understand how nonsense classifiers affect diversity of dialogue. Interesting entities include game-specific nouns (e.g. "builds", "fleet", "London", "lon", "Austria") as well as plural cardinals (e.g. "two" or "4"). No denoising or generations-based classifiers show a negative effect, whereas the token-likelihood-based and entity-corruption filters do.

Model #	**Live filter rate (%)**
1	5.48
2	0.77
3	2.65
4	18.08
5	9.47
6	17.42
7	13.64
8	16.12
9	12.31
10	16.82
11	18.77
12	1.18
13	17.04
14	4.80
15	19.71
16	7.58
Average	11.37
Ensemble	52.78

Table S5: **Nonsense ensemble filtering rates in human games** We report filtering rates for each nonsense classifier in our ensemble in live games. Descriptions of each model per model # can be found in Table S4. Many models in the final tuned ensemble contributed roughly equally, with no model dominating.

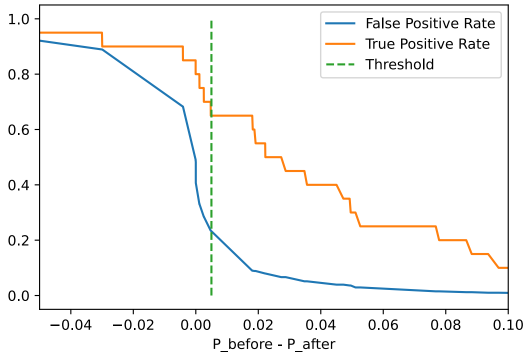

Figure S5: **Intent correspondence results** True positive rate and false positive rate for the intent correspondence filter on an expert-annotated validation set on whether the filter agreed with the expert's annotation for whether a message's dialogue was consistent with the actions in a given intent or not. "Threshold" is the threshold used in CICERO, which filtered about 65% of messages that contradicted their intents while only losing 24% of other messages.

Heuristic Filter	# Messages Filtered	% Messages Filtered
Repetitive messages	51	3.8%
Short messages	133	10.0%
Grounding	10	0.8%
Offensive/Rude language	4	0.3%
Data formatting / Other	14	1.1%
(passed all heuristic filters)	1124	84.5%

Table S6: **Heuristic message filtering statistics** The number of messages filtered by the various heuristic filters or offensive language filters implemented in CICERO, across a sample of 1330 random raw dialogue model generations among all situations encountered in the first 15 official live games against human players. Numbers sum to slightly more than 1330 or 100% due to a very small number of messages that triggered more than one filter simultaneously.

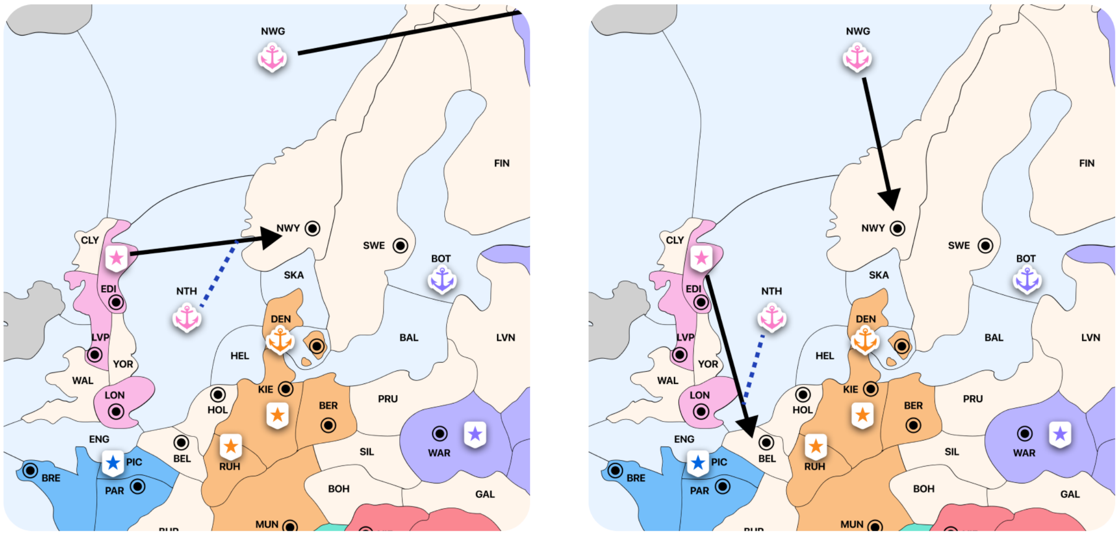

Figure S6: Vulnerability of the imitation-learned policy to manipulation through dialogue. **Left:** In this example game situation, imitation learning places about 94% likelihood on the agent, England, to convoy an army to Norway, a natural followup to England's opening. Depicted is the top-rated such action. **Right:** After Russia messages England "Thanks for agreeing to convoy your army to Bel this turn!" (which Russia would strongly prefer), even though no such agreement was ever made, the same model now places 85% likelihood on England instead convoying to Belgium. We hypothesize that the model is manipulable in this way since such messages tend to only occur in human games when there is in fact agreement, even though they are caused by the agreement rather than the cause of the agreement. Imitation learning misunderstands the causality and is exploitable as a result. CICERO's planning process eliminates this weakness.

Feature	**Type**	**Number of Channels**
Presence of army/fleet?	Binary	2
Army/fleet owner	One-hot (7 players), or all zero	7
Build turn build/disband	Binary	2
Dislodged army/fleet?	Binary	2
Dislodged unit owner	One-hot (7 players), or all zero	7
Land/coast/water	One-hot	3
Supply center owner	One-hot (7 players), or all zero	8
Home center	One-hot (7 players), or all zero	7

Table S7: Per-location board state input features for dialogue-free model

Feature	**Type**	**Number of Channels**
Number of builds allowed during winter	Float	1

Table S8: Per-player board state input features for dialogue-free model

Feature	**Type**	**Channels**
Season (spring/fall/winter)	One-hot	3
Year (encoded as $(y - 1901)/10$)	Float	1
Regular Diplomacy vs dialogue-free Diplomacy	Binary	1
Scoring system used	One-hot	2

Table S9: Global board state input features for dialogue-free model

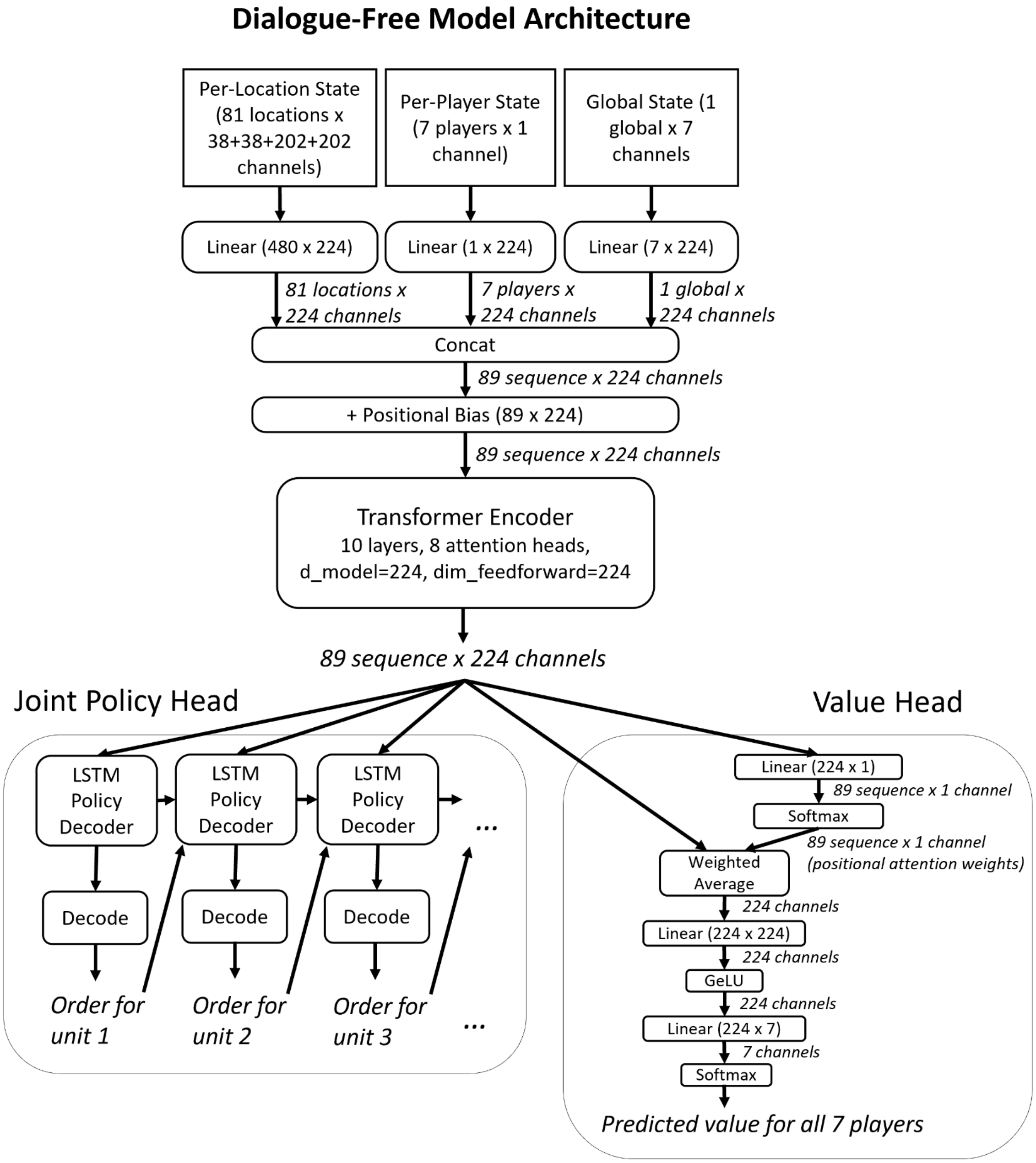
Dialogue-Free Model Architecture
Per-Location State (81 locations x 38+38+202+202 channels)
Per-Player State (7 players x 1 channel)
Global State (1 global x 7 channels
Linear (480 x 224)
Linear (1 x 224)
Linear (7 x 224)
81 locations x 224 channels
7 players x 224 channels
1 global x 224 channels
Concat
89 sequence x 224 channels
+ Positional Bias (89 x 224)
89 sequence x 224 channels
Transformer Encoder
10 layers, 8 attention heads, d_model=224, dim_feedforward=224
89 sequence x 224 channels
Joint Policy Head
LSTM Policy Decoder
LSTM Policy Decoder
LSTM Policy Decoder
...
Decode
Decode
Decode
Order for unit 1
Order for unit 2
Order for unit 3
...
Value Head
Linear (224 x 1)
89 sequence x 1 channel
Softmax
89 sequence x 1 channel (positional attention weights)
Weighted Average
224 channels
Linear (224 x 224)
224 channels
GeLU
224 channels
Linear (224 x 7)
7 channels
Softmax
Predicted value for all 7 players

Figure S7: Architecture for dialogue-free modeling of joint policy and value used for RL. Figure reproduced and updated from (*17*) that this model architecture was based on.

Parameter	**Value**
Optimization	
Use joint action as target probability	1/9
Use double joint power action as target probability	4/9
Use single power action as target probability	4/9
Batch size	500
Gradient clipping	0.5
Number GPUs	32 (V100)
Learning rate	0.002
Learning rate decay	0.99
Warmup epochs	10
Num epochs	400
Probability to add power conditioning	50%
Value head loss weight	0.5
Encoder (Transformer)	
Activation function	GeLU
Dropout	0.3
Layer size in fully connected layers	224
Number of transformer blocks	10
Number in attention heads	8
Hidden size	224
Decoder (LSTM)	
Dropout	0.3
Num layers	2
Hidden size	224
Value head	
Initialization scale multiplier	0.01
Num layers	2
Hidden size	224
Activation	GeLU
Final activation	Softmax

Table S10: Hyperparameters for behavior cloning with the dialogue-free model.

Value function learned from	H2H score vs 6 coordinating agents	Population score vs independent search agents
Independent improved BC (*26*)	45 ± 0.9	17 ± 0.9
Joint improved BC	46 ± 0.9	15 ± 0.9
RL-DiL-piKL (*17*)	47 ± 0.9	26 ± 1.1
RL-Cor-BR (CICERO)	55 ± 0.9	22 ± 1.0

Table S11: Strength and coordination-awareness of different value functions. In the first column we make the agent play against 6 BC agents, in the second column against a population of independent search agents. Independent improved BC is the improved imitation value modeling from (*26*). Joint improved BC is same, but using rollouts from a correlated joint policy rather than independent per-player policies. RL-DiL-piKL is the leading RL algorithm in dialogue-free Diplomacy (*17*) that also trains a value function assuming independent player policies and is not correlation-aware, whereas RL-Cor-BR is correlation-aware. Correlation-aware RL shows a significant increase in performance vs correlated opponents, whereas non-correlation-aware RL performs better versus independent search opponents.

Row agent policy	1 row agent vs 6 BC agent	6 row agent vs 1 BC Search agent
Independent BC (*26*)	13 ± 0.5	47 ± 0.9
Joint BC	13 ± 0.5	52 ± 0.9
RL-Cor-BR (CICERO)	20 ± 0.6	54 ± 0.9

Table S12: Evaluation of policy strength and coordination. Independent BC is trained to predict likely human player actions, Joint BC is trained to predict likely joint actions of all players. In the first column we measure how strong the policy is when controlling a single agent versus 6 BC agents that are jointly controlled by one joint BC policy. CICERO's RL policy outperforms BC policies.

In the second column we show the score when the row agent policy jointly controls 6 agents against one BC search agent. Joint BC outperforms independent BC due to modeling how the 6 controlled agents would correlate their actions with one another, CICERO's RL policy outperforms yet further due to the benefit of RL.

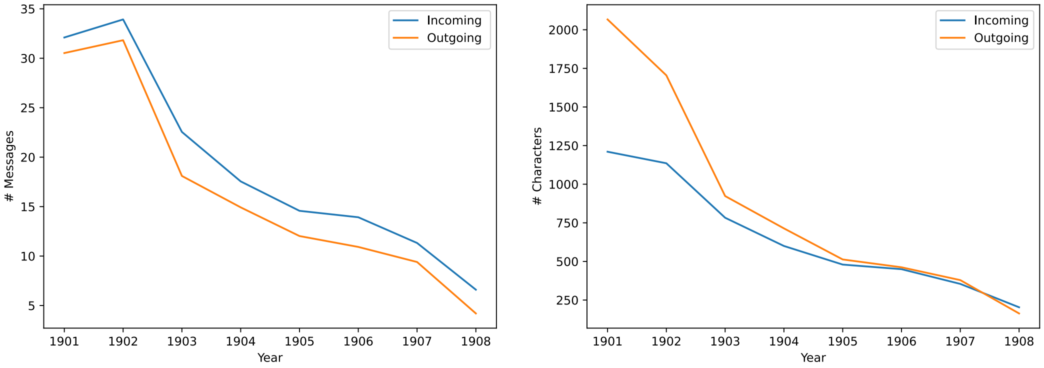

Figure S8: Number of incoming and outgoing messages from CICERO by game year, across 17 anonymous blitz Diplomacy games with human players. CICERO tends to send and receive a similar number of messages, but tends to write longer messages in 1901 and 1902.

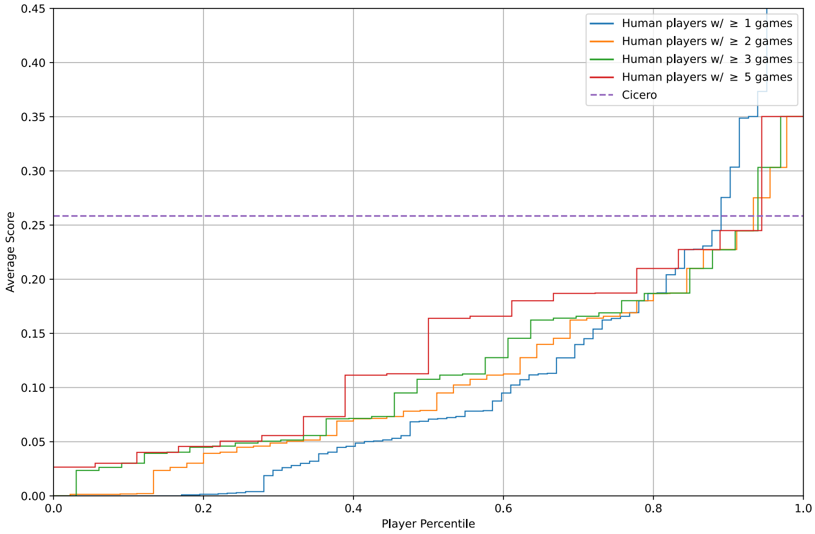

Figure S9: Cumulative histogram of player scores, restricted to those who played in at least 1, 2, 3, and 5 games in the tournament with CICERO.

Rank	Average Score	# Games Played
1	0.4918	1
2	0.4912	1
3	0.4528	1
4	0.4516	1
5	0.3734	1
6	0.3505	11
7	0.3491	1
8	0.3035	4
9	0.2754	2
10	**0.2583**	**40**
11	0.2450	6
12	0.2311	1
13	0.2277	2
14	0.2274	8
15	0.2101	5
16	0.2045	1
17	0.1875	7
18	0.1870	5
19	0.1803	8
20	0.1692	4
21	0.1659	11
22	0.1641	7
23	0.1624	3
24	0.1540	1
25	0.1454	3
26	0.1399	2
27	0.1277	4
28	0.1276	1
29	0.1132	1
30	0.1128	6
31	0.1116	7
32	0.1076	3
33	0.1025	2
34	0.0952	4
35	0.0880	1
36	0.0789	2
37	0.0786	1
38	0.0782	2
39	0.0733	9
40	0.0726	1
41	0.0716	3
42	0.0713	4
43	0.0692	2
44	0.0686	1
45	0.0559	8
46	0.0533	1
47	0.0517	3
48	0.0511	1
49	0.0506	5
50	0.0489	3
51	0.0459	5
52	0.0449	3
53	0.0404	8
54	0.0393	3
55	0.0321	1
56	0.0301	6
57	0.0283	1
58	0.0265	7
59	0.0236	3
60	0.0190	1
61	0.0042	1
62	0.0040	1
63	0.0029	1
64	0.0028	1
65	0.0021	2
66	0.0017	2
67	0.0015	2
68	0.0014	2
69	0.0014	2
70	0.0000	1
71	0.0000	1
72	0.0000	1
73	0.0000	1
74	0.0000	1
75	0.0000	1
76	0.0000	3
77	0.0000	1
78	0.0000	1
79	0.0000	1
80	0.0000	1
81	0.0000	1
82	0.0000	1
83	0.0000	1

Table S13: **Table of player scores in anonymous human games.** CICERO (**bold**) placed 10th of 83 competitors, and 2nd out of 19 who played at least 5 games.

	xplayer	effective	xplayer moves	coordinated
CICERO	6.875	1.475	4.075	2.600
CICERO (start of phase)	5.775	1.100	3.550	2.075
Diplodocus-High (*17*)	7.375	0.975	3.625	1.575
Human players	6.921	1.721	4.800	3.514

Table S14: Comparing cross-player support effectiveness and coordination for different (counterfactual) sets of orders in human games. CICERO played on the webDiplomacy Blitz league. All numbers are average counts per game. CICERO (start of phase) refers to CICERO's order generation run at the beginning of the phase, before any dialogue. Diplodocus-High is the state-of-the-art agent for no-press Diplomacy. Human players refers to the other players in the games where CICERO played. **xplayer**: The total number of cross-player supports by the agent. **effective**: The percentage of cross-player supports to or from the agent that were 'effective', as defined in (*7*), i.e. that the outcome would have been different if the support had been changed to a hold. **xplayer moves**: The total number of cross-player support moves by the agent. **coordinated**: The total number of cross-player support moves to or from the agent for which the supported move was actually played.

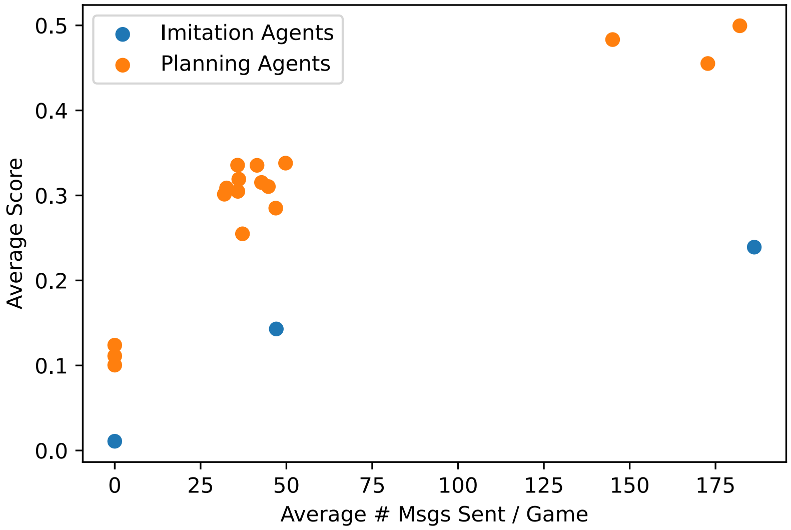

Figure S10: Scatter plot showing the relationship between number of messages sent and agent performance, against 6x imitation agents (scores from Table S15). The number of messages sent dominates the effect of other dialogue factors against this population. This suggests testing versus imitation agents may not be a reliable way of evaluating the effect of changes in dialogue generation on the quality of the final agent.

	6x Agent	
1x Agent	Imitation Agent	CICERO
CICERO	$32.0\% \pm 1.7\%$	$14.3\% \pm 0.0\%$
w/o intents	$35.5\% \pm 1.8\%$	$14.5\% \pm 1.1\%$
w/o nonsense filters	$35.8\% \pm 1.8\%$	
w/o dialogue	$13.7\% \pm 1.2\%$	
w/o dialogue, w/o dialogue conditioning	$10.8\% \pm 1.1\%$	$10.5\% \pm 1.0\%$
w/ SL value function	$25.8\% \pm 1.5\%$	$8.2\% \pm 0.8\%$
w/ talkative heuristics	$50.4\% \pm 1.9\%$	$15.7\% \pm 1.2\%$
w/ talkative heuristics, w/o intents	$52.0\% \pm 1.9\%$	
Imitation Agent	$14.3\% \pm 0.0\%$	$4.5\% \pm 0.5\%$
w/o dialogue	$1.1\% \pm 0.3\%$	
w/ talkative heuristics	$23.6\% \pm 1.6\%$	$4.8\% \pm 0.5\%$
Diplodocus-High (*17*)	$11.3\% \pm 1.0\%$	

Table S15: Average score of 1x agent in head-to-head games against six copies of 6x agent in simulated Blitz Diplomacy games with an end year of 1908. Since there are 7 players, note that equal performance of the 1x agent would be $1/7 \approx 14.3\%$. **w/o intents:** Intents are not computed. A dialogue model that doesn't condition on intents is used. All filtering is disabled. **w/o nonsense filters:** All nonsense filters are disabled. **w/o dialogue**: The agent does not generate dialogue. **w/o dialogue conditioning**: Dialogue is ignored during planning. A dialogue-free model (§E.3) is used for the anchor policy for planning. **w/ talkative heuristics**: We enable heuristics used during Blitz evaluations, in which we compute a delay time for a message but then set it to 15 seconds as long as the message will be sent this turn. This leads to about 4x more messages sent per game. We disable this heuristic in the base CICERO agent for a more fair comparison with the imitation agent.

References and Notes

1. T. Brown *et al*., Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877 (2020).

2. M. Campbell, A. J. Hoane Jr., F. Hsu, Deep Blue. *Artif. Intell.* **134**, 57–83 (2002).

3. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

4. N. Brown, T. Sandholm, Superhuman AI for multiplayer poker. *Science* **365**, 885–890 (2019).

5. S. Kraus, D. Lehmann, Diplomat, an agent in a multi agent environment: An overview, in *IEEE International Performance Computing and Communications Conference* (IEEE Computer Society, 1988), pp. 434–435.

6. D. d. Jonge *et al*., The challenge of negotiation in the game of *Diplomacy*, in *International Conference on Agreement Technologies* (Springer, 2018), pp. 100–114.

7. P. Paquette *et al*., No-press *Diplomacy*: Modeling multi-agent gameplay. *Adv. Neural Inf. Process. Syst.* **32**, 4474–4485 (2019).

8. A. Dafoe, Y. Bachrach, G. Hadfield, E. Horvitz, K. Larson, T. Graepel, Cooperative AI: Machines must learn to find common ground. *Nature* **593**, 33–36 (2021).

9. M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* **356**, 508–513 (2017).

10. N. Brown, T. Sandholm, Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* **359**, 418–424 (2018).

11. O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, Grandmaster level in *StarCraft II* using multi-agent reinforcement learning. *Nature* **575**, 350–354 (2019).

12. Dota 2 (*32*) is two-team zero-sum but with unlimited information sharing between teammates, which makes the game equivalent to 2p0s. Prior work found that self-play from scratch was sufficient for achieving superhuman performance in multiplayer poker (*4*), but this may be due to poker offering few opportunities for players to cooperate.

13. J. v. Neumann, Zur theorie der gesellschaftsspiele. *Math. Ann.* **100**, 295 (1928).

14. M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, D. Batra, Deal or no deal? End-to-end learning of negotiation dialogues, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Copenhagen, Denmark, 2017), pp. 2443–2453.

15. A. P. Jacob, M. Lewis, J. Andreas, Multitasking inhibits semantic drift, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021), pp. 5351–5366.

16. A. Bakhtin, D. Wu, A. Lerer, N. Brown, No-press *Diplomacy* from scratch. *Adv. Neural Inf. Process. Syst.* **32**, 34 (2021).

17. A. Bakhtin *et al*., Mastering the game of no-press *Diplomacy* via human-regularized reinforcement learning and planning. arXiv:2210.05492 [cs.GT] (2022).

18. A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi, The curious case of neural text degeneration, *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* (OpenReview.net, 2020).

19. S. Keizer *et al*., Evaluating persuasion strategies and deep reinforcement learning methods for negotiation dialogue agents, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (Association for Computational Linguistics, 2017), pp. 480–484.

20. T. Hiraoka, G. Neubig, S. Sakti, T. Toda, S. Nakamura, Construction and analysis of a persuasive dialogue corpus, in *Situated Dialog in Speech-Based Human-Computer Interaction* (Springer, 2016), pp. 125–138.

21. X. Wang *et al*., Persuasion for good: Towards a personalized persuasive dialogue system for social good, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics, Florence, Italy, 2019), pp. 5635–5649.

22. K. Shuster *et al*., Language models that seek for knowledge: Modular search and generation for dialogue and prompt completion. arXiv:2203.13224 (2022).

23. A. Vaswani *et al*., Attention is all you need, in *Advances in Neural Information Processing Systems*, I. Guyon, *et al.*, eds. (Curran Associates, Inc., 2017), vol. 30.

24. M. Lewis *et al*., BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5–10, 2020*, D. Jurafsky, J. Chai, N. Schluter, J. R. Tetreault, Eds. (Association for Computational Linguistics, 2020), pp. 7871–7880.

25. N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, R. Socher, CTRL: A conditional transformer language model for controllable generation. arXiv:1909.05858 [cs.CL] (2019).

26. A. P. Jacob *et al*., Modeling strong and human-like gameplay with KL-regularized search, in *International Conference on Machine Learning* (PMLR, 2022), pp. 9695–9728.

27. D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362**, 1140–1144 (2018).

28. N. Brown, A. Bakhtin, A. Lerer, Q. Gong, Combining deep reinforcement learning and search for imperfect-information games. *Adv. Neural Inf. Process. Syst.* **33**, 17057 (2020).

29. Z. Ji, *et al.*, Survey of hallucination in natural language generation. arXiv:2202.03629 [cs.CL] (2022).

30. P. Gupta, Y. Tsvetkov, J. P. Bigham, Synthesizing adversarial negative responses for robust response ranking and evaluation, in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, R. Navigli, Eds. (Association for Computational Linguistics, 2021), vol. ACL/IJCNLP 2021 of *Findings of ACL*, pp. 3867–3883.

31. M. Alzantot *et al*., Generating natural language adversarial examples, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31–November 4, 2018,* E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii, eds. (Association for Computational Linguistics, 2018), pp. 2890–2896.

32. C. Berner *et al*., Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680 (2019).

33. FAIR *et al.*, Supplementary data for "Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning". Zenodo (2022); doi:10.5281/zenodo.7236700.

34. FAIR *et al*., Code for "Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning". GitHub (2022); https://github.com/facebookresearch/diplomacy_cicero.

35. N. Carlini *et al*., Extracting training data from large language models, in *30th USENIX Security Symposium (USENIX Security 21)* (2021), pp. 2633–2650.

36. L. Weidinger, *et al.*, Ethical and social risks of harm from language models. arXiv:2112.04359v1 [cs.CL] (2021).

37. E. M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, On the dangers of stochastic parrots: Can language models be too big? *Proc. FAccT* **2021**, 610–623 (2021).

38. E. Dinan, *et al.*, Anticipating safety issues in E2E conversational AI: Framework and tooling. arXiv:2107.03451 [cs.CL] (2021).

39. I. Gabriel, Artificial intelligence, values, and alignment. *Minds Mach.* **30**, 411–437 (2020).

40. A. Bakhtin, *et al.*, Real or fake? Learning to discriminate machine from human generated text. arXiv:1906.03351 [cs.LG] (2019).

41. R. Zellers *et al*., Defending against neural fake news, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, H. M. Wallach, *et al.,* eds. (2019), pp. 9051–9062.

42. C. Governor, California new autobot law, cal. bus. and prof. code § 17940, et seq. (sb 1001) (2018).

43. D. J. H. Burden, M. Savin-Baden, R. Bhakta, Covert implementations of the Turing test: A more level playing field?, in *Research and Development in Intelligent Systems XXXIII*, M. Bramer, M. Petridis, Eds. (Springer International Publishing, 2016), pp. 195–207.

44. L. Clark *et al*., What makes a good conversation?: Challenges in designing truly conversational agents, in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04–09, 2019*, S. A. Brewster, G. Fitzpatrick, A. L. Cox, V. Kostakos, Eds. (ACM, 2019), p. 475.

45. W. Shi *et al*., Effects of persuasive dialogues: Testing bot identities and inquiry strategies, *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25–30, 2020*, R. Bernhaupt, *et al*., Eds. (ACM, 2020), pp. 1–13.

46. W. Shi, Y. Li, S. Sahay, Z. Yu, Refine and imitate: Reducing repetition and inconsistency in persuasion dialogues via reinforcement learning and human demonstration, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, M. Moens, X. Huang, L. Specia, S. W. Yih, Eds. (Association for Computational Linguistics, 2021), pp. 3478–3492.

47. S. Kraus, E. Ephrati, D. Lehmann, Negotiation in a non-cooperative environment. *J. Exp. Theor. Artif. Intell.* **3**, 255–281 (1994).

48. S. Kraus, D. Lehmann, Designing and building a negotiating automated agent. *Comput. Intell.* **11**, 132–171 (1995).

49. S. J. Johansson, F. Håård, Tactical coordination in no-press *Diplomacy*, in *International Joint Conference on Autonomous Agents and Multiagent Systems* (2005), pp. 423–430.

50. A. Ferreira, H. L. Cardoso, L. P. Reis, Dipblue: A *Diplomacy* agent with strategic and trust reasoning, in *ICAART 2015—Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 1, Lisbon, Portugal, 10–12 January, 2015*, S. Loiseau, J. Filipe, B. Duval, H. J. van den Herik, Eds. (SciTePress, 2015), pp. 54–65.

51. J. Marinheiro, H. Lopes Cardoso, Towards general cooperative game playing, in *Transactions on Computational Collective Intelligence XXVIII* (Springer, 2018), pp. 164–192.

52. J. van Hal, Diplomacy AI - Albert (2013); https://sites.google.com/site/diplomacyai.

53. J. Gray, A. Lerer, A. Bakhtin, N. Brown, Human-level performance in no-press Diplomacy via equilibrium search, in *International Conference on Learning Representations* (2020).

54. T. Anthony *et al*., Learning to play no-press *Diplomacy* with best response policy iteration. *Adv. Neural Inf. Process. Syst.* **33**, 17987 (2020).

55. D. Peskov, B. Cheng, It takes two to lie: One to lie, and one to listen, in *Proceedings of ACL* (2020).

56. N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, M. Bowling, The Hanabi challenge: A new frontier for AI research. *Artif. Intell.* **280**, 103216 (2020).

57. M. Carroll *et al.*, On the utility of learning about humans for human-AI coordination. *Adv. Neural Inf. Process. Syst.* **32**, 5174–5185 (2019).

58. D. Strouse, K. McKee, M. Botvinick, E. Hughes, R. Everett, Collaborating with humans without human data. *Adv. Neural Inf. Process. Syst.* **34**, 14502 (2021).

59. A. Lerer, A. Peysakhovich, Learning existing social conventions via observationally augmented self-play, in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (ACM, 2019), pp. 107–114.

60. J. Nash, Non-cooperative games. *Ann. Math.* **54**, 286–295 (1951).

61. S. Hart, A. Mas-Colell, A simple adaptive procedure leading to correlated equilibrium. *Econometrica* **68**, 1127–1150 (2000).

62. E. Levin, R. Pieraccini, W. Eckert, A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans. Speech Audio Process.* **8**, 11–23 (2000).

63. S. Young, M. Gašić, B. Thomson, J. D. Williams, POMDP-based statistical spoken dialog systems: A review, in *Proceedings of the IEEE* **101** (2013).

64. J. D. Williams, K. Asadi, G. Zweig, Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017).

65. H. He, D. Chen, A. Balakrishnan, P. Liang, Decoupling strategy and generation in negotiation dialogues, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Brussels, Belgium, 2018), pp. 2333–2343.

66. J. Schatzmann, K. Weilhammer, M. Stuttle, S. Young, A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl. Eng. Rev.* **21**, 97–126 (2006).

67. V. Rieser, O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems: A Data-Driven Methodology for Dialogue Management and Natural Language Generation* (Springer, 2011).

68. D. Yarats, M. Lewis, Hierarchical text generation and planning for strategic dialogue, in *International Conference on Machine Learning* (PMLR, 2018), pp. 5591–5599.

69. D. Traum, S. C. Marsella, J. Gratch, J. Lee, A. Hartholt, Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents, *International Workshop on Intelligent Virtual Agents* (Springer, 2008), pp. 117–130.

70. I. Efstathiou, O. Lemon, Learning non-cooperative dialogue behaviours, in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)* (Association for Computational Linguistics, Philadelphia, PA, U.S.A., 2014), pp. 60–68.

71. K. Chawla *et al.*, CaSiNo: A corpus of campsite negotiation dialogues for automatic negotiation systems, in *Proceedings of the 2021 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies (Association for Computational Linguistics, Online, 2021), pp. 3167–3185.

72. Y. Li, K. Qian, W. Shi, Z. Yu, End-to-end trainable non-collaborative dialog system. *Proc. Conf. AAAI Artif. Intell.* **34**, 8293–8302 (2020).
73. Y. Tian, W. Shi, C. Li, Z. Yu, Understanding user resistance strategies in persuasive conversations, in *Findings of the Association for Computational Linguistics: EMNLP 2020* (Association for Computational Linguistics, 2020), pp. 4794–4798.
74. S. Afantenos *et al*., Developing a corpus of strategic conversation in The Settlers of Catan, *Workshop on Games and NLP (GAMNLP-12)* (2012).
75. H. Cuayáhuitl, S. Keizer, O. Lemon, Strategic dialogue management via deep reinforcement learning, in *NeurIPS Workshop on Deep Reinforcement Learning* (2015).
76. J. Andreas, D. Klein, Reasoning about pragmatics with neural listeners and speakers, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, 2016), pp. 1173–1182.
77. W. Monroe, R. X. Hawkins, N. D. Goodman, C. Potts, Colors in context: A pragmatic neural model for grounded language understanding. *Trans. Assoc. Comput. Linguist.* **5**, 325–338 (2017).
78. A. Radford *et al.*, Language models are unsupervised multitask learners. *OpenAI blog* **1**, 9 (2019).
79. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio, Y. LeCun, eds. (2015).
80. A. H. Miller *et al*., ParlAI: A dialog research software platform, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017 - System Demonstrations*, L. Specia, M. Post, M. Paul, Eds. (Association for Computational Linguistics, 2017), pp. 79–84.
81. E. M. Smith, D. Gonzalez-Rico, E. Dinan, Y. Boureau, Controlling style in generated dialogue. arXiv:2009.10855 [cs.CL] (2020).
82. J. Xu, *et al.*, Recipes for safety in open-domain chatbots. arXiv:2010.07079 [cs.CL] (2020).
83. J. Wei, *et al.*, Chain of thought prompting elicits reasoning in large language models. arXiv:2201.11903 [cs.CL] (2022).
84. A. Fan, M. Lewis, Y. N. Dauphin, Hierarchical neural story generation, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych, Y. Miyao, eds. (Association for Computational Linguistics, 2018), pp. 889–898.
85. S. Gehman, S. Gururangan, M. Sap, Y. Choi, N. A. Smith, RealToxicityPrompts: Evaluating neural toxic degeneration in language models, in *Findings of the Association for Computational Linguistics: EMNLP 2020* (Association for Computational Linguistics, 2020), pp. 3356–3369.

86. D. Traum, Issues in multiparty dialogues, *Workshop on Agent Communication Languages* (Springer, 2003), pp. 201–211.

87. H. Ouchi, Y. Tsuboi, Addressee and response selection for multi-party conversation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Austin, Texas, 2016), pp. 2133–2143.

88. N. Littlestone, M. K. Warmuth, The weighted majority algorithm. *Inf. Comput.* **108**, 212–261 (1994).

89. N. Brown, C. Kroer, T. Sandholm, Dynamic thresholding and pruning for regret minimization, *Proceedings of the AAAI Conference on Artificial Intelligence* (2017), vol. 31.

90. D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).