Université de Montréal

Faculté des études supérieures et postdoctorales

Ce mémoire intitulé

No Press Diplomacy

présenté par

Philip Paquette

a été évalué par un jury composé des personnes suivantes :

Liam Paull
(président-rapporteur)

Aaron Courville
(directeur de recherche)

Pierre Poulin
(membre du jury)

Mémoire accepté le :
31 août 2019

Sommaire

Ce mémoire présente un article sur un agent pouvant jouer à la version "No-Press" (sans messages) du jeu de société Diplomacy. Diplomacy est un jeu de négociation à 7 joueurs où chacun des joueurs essaie de conquérir la majorité des centres d'approvisionnement d'Europe au début du 20e siècle.

L'article présente, en premier lieu, un ensemble de données contenant plus de 150 000 jeux joués par des humains. Cet ensemble de données a été compilé suite à la signature d'un partenariat avec un site externe. Les jeux, qui ont été joués sur cette plateforme, ont tous été convertis dans un nouveau format standardizé et ont ensuite été rejoués pour s'assurer de leur qualité. L'article présente aussi un engin de jeu, avec une interface web, permettant à des humains de jouer contre les modèles qui ont été développés.

De plus, l'article présente un modèle d'apprentissage supervisé où l'agent apprend à reproduire le comportement de tous les joueurs dans l'ensemble de données par maximum de vraisemblance. Un agent qui apprend à jouer par renforcement (en jouant contre lui-même) a aussi été entraîné. L'article se conclut en faisant une analyse de ces modèles et en comparant la performance des agents contre des agents utilisant des règles complexes.

Mot-clés: Diplomacy, négociation, jeu, apprentissage supervisé, apprentissage par renforcement, apprentissage profond

Summary

This thesis presents an article on an agent which can play the "No-Press" version (without messages) of the Diplomacy board game. Diplomacy is a 7-player negotiation game where each player tries to conquer the majority of the supply centers in Europe at the beginning of the 20th century.

The article first presents a novel dataset of more than 150 000 human games. This dataset was compiled following the signing of a partnership with an external site. The games, which were played on this platform, were all converted into a new standardized format and then replayed to ensure their quality. The article also presents a game engine, with a web interface, allowing humans to play against the models that have been trained.

Moreover, the article presents a supervised learning model where an agent learns to reproduce the behavior of all players in the dataset by maximum likelihood. An agent that learns by reinforcement (by playing games against itself) has also been trained. The article concludes by doing an analysis of these models and comparing their performance against complex rule-based agents.

Keywords: Diplomacy, negotiation, game, supervised learning, reinforcement learning, deep learning

Table of Contents

List of tables

List of figures

Chapter 1

Introduction

Diplomacy is a negotiation board game that was developed in 1959 by Allan B. Calhamer. Seven powers, namely Austria, England, France, Germany, Italy, Russia, and Turkey, try to conquer a majority of the supply centers of Europe at the beginning of the 20th century. Diplomacy is an interesting domain for artificial intelligence for several reasons:

- **Large action space**. There can be up to 34 units on the board (one for each supply center), with each unit having an average of 26 possible actions, giving an average branching factor of $1.29 * 10^{48}$. Compared to Go (branching factor of 200 [**5**]), or chess (branching factor of 35 [**5**]), Diplomacy is several orders of magnitude more complex, making techniques such as search and rollout much more difficult to implement.

- **No randomness**. As opposed to games like Risk [**21**] (which uses dice) or Hanabi [**3**] (which uses shuffled cards), Diplomacy has no elements of randomness and is purely a skills game.

- **Negotiation**. Players want to act in their best interests, yet to win the game, they must also negotiate and cooperate with other players. Negotiation can be done either explicitly through messages, or implicitly through the orders that players submit simultaneously. Alliances between players are usually short-lived and betrayal and deception is common.

- **Language emergence and grounding**. Diplomacy can be an interesting testbed for language emergence. For instance, are agents able to create a new language among themselves to play better than agents that are not able to communicate? Moreover, the conversations need to be grounded in the game context, otherwise the players will not be able to understand each other.

- **Imperfect and incomplete information**. Orders are submitted by all players simultaneously, therefore players are not aware of what other players will do when deciding what orders to play. Moreover, messages exchanged between players are private, and can only be seen by its sender and recipient. Finally, the players' strategy are not fully known and may change during the game.

- **Coalition formation**. A successful strategy usually requires that players collaborate with other players to strengthen their attacks or defenses. Coalition formation is a key component of the game, even though alliances are usually short-lived.

- **Machine theory of mind**. Players need to reason over the intentions of their opponents. Is this person genuinely interested in helping me, is he lying, or is it part of a bigger strategy where I might risk being betrayed? Trust reasoning, where one evaluates if others are trustworthy or not, is also a key component of understanding the intention of others.

- **Multiple (and long) time scales**. Players need to be able to form a long-term strategy on how they plan on winning the game, and a short-term execution strategy that fits within their bigger plan. Players must also be able to adapt their plans quickly if the game unfolds in a unexpected direction.

- **Cooperating and competitive**. Diplomacy requires both cooperation and competition. Players are self-interested, but need to build short-term alliances

to reach their long-term objectives. This mix of cooperative and competitive environments makes computing the value of a board state much more difficult, because a player that is almost eliminated, but has strong allies might still survive and win the game (or at least be part of a draw).

- **Simultaneous communications**. Players must learn to have simultaneous dialogues between different powers and take into account what was said in other conversations to have consistent and meaningful dialogues.

This thesis only focuses on the No Press version of Diplomacy (without messages), therefore some of these challenges have not been explored. Even though there are no messages, players can still negotiate and signal each order through their orders.

Chapter 2

Machine Learning Basics

In this chapter, I will briefly introduce the machine learning concepts required to understand the article presented thereafter. The goal of the article is to develop an agent (i.e. a bot) that can learn to play the game Diplomacy. What is the meaning of learning in this context? An agent learns to perform some task T if its performance P improves after getting some experience E.

For Diplomacy, the task T is playing the game, the performance P can be either the log likelihood, the accuracy of the orders on the validation set, or the average number of supply centers at the end of a game, and the experience E are the human games used to train the model.

2.1. Supervised Learning

In supervised learning, a dataset of points (x, y) is used for learning. x is the input to the model (i.e. the board state for Diplomacy) and y is the label or target (i.e. the human orders). The supervised algorithm is trying to learn a model that maximizes the conditional probability $p(y|x)$, and can therefore predict, to the best of its ability, a sequence of orders (i.e. one order per unit) that human players would play in that particular situation.

A player needs to submit simultaneously one order for each unit it has on the board. For instance, at the beginning of the game, France has a fleet in Brest (F BRE), an army in Paris (A PAR), and an army in Marseilles (A MAR). The model needs to be able to predict all three locations simultaneously. Coordination is very important among

the units, because what one might do in Marseilles will likely depend on what one does in Paris.

The supervised algorithm needs to be able to predict the joint probability of the orders across all locations given the board state, namely $p(y_1, y_2, ..., y_n|x)$. Using the chain rule, we can break down this probability as:

$$p(y_1, y_2, ..., y_n|x) = p(y_1|x)p(y_2|y_1, x)...p(y_n|y_1, y_2, ..., y_{n-1}, x)$$

where $y_1 = BRE$, $y_2 = PAR$, $y_3 = MAR$ in the above example. We can model this joint probability distribution using a recurrent neural network, as discussed later.

2.1.1. Overfitting and Underfitting

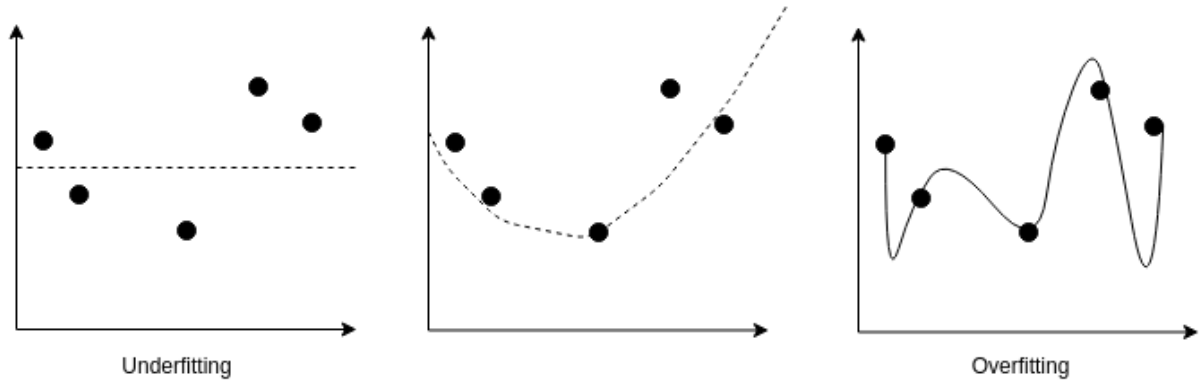


Fig. 2.1. Underfitting vs Overfitting.

One important concept in machine learning is generalization. Generalization is the ability to predict the correct output on data points drawn from the same distribution as the training set that were not used to train the model. For instance, in Diplomacy, we want the model to be able to predict the correct human orders, even on a board state that the model has never seen.

To be able to measure the generalization error, the dataset is usually split in three parts, namely the training set, the validation set, and the test set. The model is trained using the training set, the hyper-parameters are tuned using the validation test, and the generalization error is computed using the test set.

Models can also be regularized, by adding modifications that have the goal of reducing the generalization error on the test set, but not the training error. An example of regularization is dropout **[35]**, where a certain proportion of inputs and hidden layer neurons are dropped during training. The capacity of the model can be controlled using the hyper-parameters. For instance, the number of hidden layers is one example of a hyper-parameter.

We want to choose hyper-parameters that give a good generalization error. If the capacity of the model is too low, the model will not be able to represent the target distribution and will likely have a high error on both the training and test set (i.e. underfitting scenario above). If the capacity of the model is too great, the training error will likely be very small, but the model probably will not be able to generalize well to unseen examples because it learned the noise on the data (i.e. overfitting scenario above). We want to choose a set of hyper-parameters that gives an appropriate capacity for the model, and that minimizes the loss calculated on the validation set.

2.1.2. Maximum Likelihood Estimation

To train our model, we want to maximize the conditional likelihood of the model, which correlates to predicting the correct human orders for each point in the training set. This corresponds to the minimization of the KL divergence:

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\theta} P(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\theta})$$

$$= \arg\max_{\theta} \prod_{i=1}^{m} p_{\text{ model}} \left(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} log\ p_{\text{ model}}\ \left(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}\right)$$

$$\approx \arg\max_{\boldsymbol{\theta}}\ \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{ data}}}\ \log p_{\text{ model}}\left(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}\right)$$

$$= -\arg\min_{\boldsymbol{\theta}}\ \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{ data}}}\ \log p_{\text{ model}}\left(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}\right)$$

Maximizing the likelihood corresponds to minimizing the KL divergence between $\hat{p}_{\text{ data}}$ and $\hat{p}_{\text{ model}}$, which also corresponds to minimizing the cross-entropy.

2.1.3. Deep Feedforward Networks

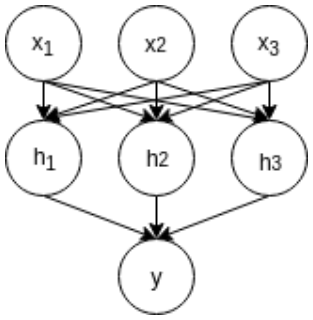


Fig. 2.2. Deep Feedforward Network / Neural Network

The feedforward neural network is the simplest neural network, where information flows from the inputs (x) to intermediate layers (h) and then to the outputs (y). The hidden layer is also called a fully-connected layer, because each node in the previous layer (i.e. x) is connected to every node in the current layer (h).

The value of h_1 can be computed as follows: $h_1 = \text{ReLu}(x_1w_1 + x_2w_2 + x_3w_3 + c)$. The entire neural network above can be written in matrix form using: $\hat{y} = w_2^T h_1 + b$. It is important to use a non-linear activation after each hidden layer, to break the linearity of the network. The rectified linear activation function (ReLU) is one of the most commonly used activation function, and can be expressed as $\max(0, x)$.

2.1.4. Stochastic Gradient Descent

Most loss functions of a neural network with non-linearities are non-convex. To optimize a non-convex function, stochastic gradient descent is commonly used.

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\hat{p}_{\text{ data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)$$

$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{m'} L\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\boldsymbol{g}$$

where L is the loss per example $L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log\, p(y|\boldsymbol{x}; \boldsymbol{\theta})$.

Gradient descent computes the gradient of the loss with respect to each parameter, and then takes a step of size ϵ in the opposite direction of the gradient (i.e. to minimize the loss). Gradient descent is computationally expensive, because it computes the loss over all the points in the dataset. More commonly, a mini-batch of size m' is used, and the loss is only computed over the points in that mini-batch. If the mini-batch has a size of 1, the procedure is referred to as stochastic gradient descent.

2.1.5. Convolutional Neural Networks

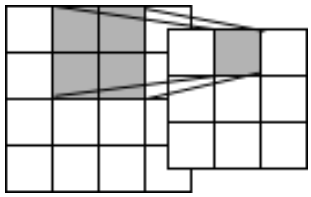


Fig. 2.3. Convolutional Neural Network

A convolutional neural network (CNN) is a neural network specialized for grid-like inputs (e.g. images). It uses a kernel that is applied repeatedly to the different regions of the image. CNNs have several advantages over fully-connected layers, namely sparse interactions, tied

weights, and equivariance to translation. Sparse interactions refer to the fact that the kernel is smaller than the inputs, therefore not all outputs are connected to all inputs, making CNNs much more computationally efficient than fully-connected layers. Tied weights refer to the fact that the same kernel is applied to multiple locations, therefore reducing the number of parameters a CNN has. Equivariance to translation refers to the fact that applying a translation before doing a convolution is similar to doing the convolution then applying the translation.

2.1.6. Recurrent Neural Networks

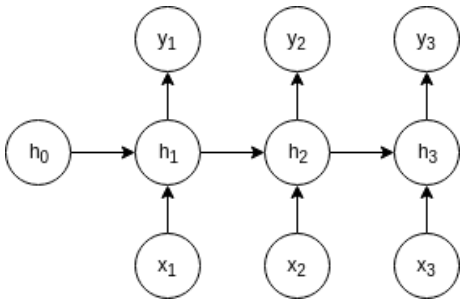


Fig. 2.4. Recurrent Neural Network

Recurrent neural networks are a specialized type of neural networks for sequential data. RNNs can scale to long sequences by sharing the same parameters for every time step. They also have a recurrence, where hidden layer h_t takes as input its previous value h_{t-1}. One of the main difficulties with RNNs is learning long-term dependencies. Applying the same weight matrix at every time step (with a recurrence in the form of $\boldsymbol{h}^{(t)} = \boldsymbol{W}^\top \boldsymbol{h}^{(t-1)}$) is equivalent to raising the eigenvalues of the weight matrix to the power of the number of time steps (i.e. $\boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^t\right)^\top \boldsymbol{h}^{(0)} = \boldsymbol{Q}^\top \boldsymbol{\Lambda}^t \boldsymbol{Q} \boldsymbol{h}^{(0)}$). Eigenvalues larger than 1 converge to infinity, while eigenvalues smaller than 1 converge to 0. RNNs have therefore a lot of difficulty remembering information over a large number of time steps because of this problem.

One common solution is to use a Long Short-Term Memory (LSTM) network. LSTMs are able to remember information over long periods of time by using a context, that is passed between time steps. There is no non-linear activation or weights applied to the context. Information can be added or removed from the context by using the input and forget gates. The LSTM is then able to decide what to output at every time step from the context by using an output gate.

2.1.7. Attention Mechanisms

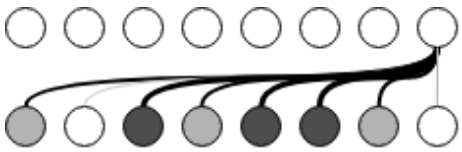


Fig. 2.5. Attention Mechanism

Attention mechanisms [**2**] allow a hidden layer to compute how much it wants to attend to each of the previous time step in a memory (using a softmax) and then do a weighted linear combination of those memory time steps. With attention, a network can decide to attend more heavily on important information that happened many time steps in the past, and therefore bypass the long-term dependencies problem that recurrent neural networks have.

2.2. Reinforcement Learning

Reinforcement learning is a method where an agent interacts with an environment, and learns to improve its behaviour over time. As shown in Figure 2.6, an agent performs an action in an environment, and then receives a reward and the next state. In Diplomacy, the action is the orders for all the units of a power, the reward is, for example, the number of supply centers conquered during the turn, and the next state is the new board state after all players have moved.

2.2.1. Markov Decision Process

A Markov Decision Process (MDP) is a tuple (S, A, P, R) that provides a framework to analyze reinforcement learning. S refers to a finite set of states in which the agent can be, A is a finite set of actions that the agent can perform in those states, $P_a(s, s')$ represents the probability of a transition from state s to state s' when action a is performed in state s, and $R_a(s, s')$ is the immediate reward that can be expected by the agent after performing action a in state s when the next state is s'.

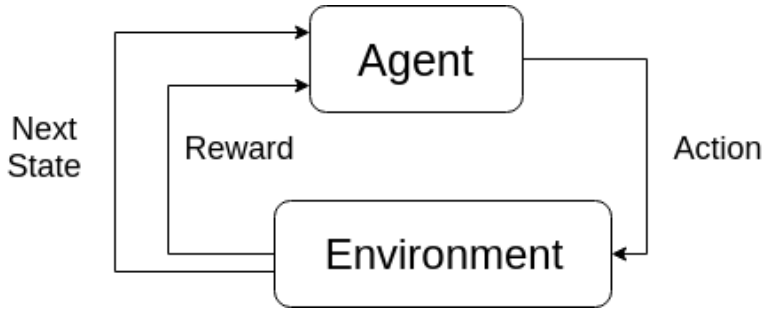


Fig. 2.6. Agent-Environment Interaction

The goal of the agent is to maximize the sum of the rewards it receives over all time steps, namely $G_t = R_{a0}(s_0, s_1) + R_{a1}(s_1, s_2) + ... = \sum_{t=0}^{T} R_{a_t}(s_t, s_{t+1})$. To have convergence, we need to have a fixed number of time steps or use a discount factor γ. The sum of discounted reward becomes $G_t = \sum_{t=0}^{T} \gamma^t R_{a_t}(s_t, s_{t+1})$.

The rewards received by an agent are dependent on the actions the agent performed. The ultimate goal of a reinforcement learning agent is to learn a policy that maximizes the sum of the discounted reward the agent will receive. There is a trade-off between exploring new states to hopefully find a better policy and get more rewards in the future, and exploiting where the agent executes its current policy to maximize the immediate reward.

2.2.2. Value Function and Q-Values

The value of a state $V(s)$ (under policy π) is the expected sum of discounted rewards an agent is expected to receive from state s if the agent executes policy π. $V_\pi(s) = \sum_{s'} P_{\pi(s)}(s, s')(R_{\pi(s)}(s, s') + \gamma V_\pi(s'))$. The q-value of a state $Q_\pi(s,a)$ is the expected sum of discounted rewards an agent is expected to receive if it starts in state s, then performs action a, and follows policy π thereafter $Q_\pi(s, a) = \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V_\pi(s'))$.

2.2.3. Solving the RL Problem - Policy Iteration

The optimal policy should follow the action that maximizes the $Q(s,a)$ value at every time step, namely $\pi(s) := \text{argmax}_a\, Q(s,a) = \text{argmax}_a \left\{\sum_{s'} P_a\left(s, s'\right)\left(R_a\left(s, s'\right) + \gamma V\left(s'\right)\right)\right\}$. A simple algorithm to solve the reinforcement learning problem is policy iteration, which iteratively computes the value of all states (i.e. value update), and then computes the best action to take at every state given those new values (i.e. policy update). The algorithm stops when the policy has converged.

2.2.4. Bootstrapping

Bootstrapping refers to using the value of a future state in the computation of the value of the current state. For instance, the two-step return can be written as $V(s) = R(s, s') + \gamma R(s', s'') + \gamma^2 V(s'')$. The Monte Carlo returns refer to the computation of the value function with no bootstrapping (i.e. by summing up all the discounted rewards received on a trajectory).

2.2.5. Solving the RL Problem - Policy Gradient

The RL problem can also be solved using a gradient step (policy gradient method).

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{a}_{i,t} | \mathbf{s}_{i,t} \right) \right) \left(\sum_{t=1}^{T} r \left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t} \right) \right)$$
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

The intuition is that we want to maximize the likelihood of actions that gave us a good reward. Performing a lot of those small updates should improve our policy over time. REINFORCE (plain policy gradients) suffers from high variance. A common variance-reduction technique is to subtract a fixed baseline (e.g. moving average of returns across all states). Another common technique is to use an actor-critic method, where the log probability of the action is multiplied by the advantage (Q-value - value of the state). The intuition is that we have a critic that can compute the value of an estimate and we want to maximize the likelihood of actions that gave us more rewards than the critic anticipated. The gradient formula becomes:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(\mathbf{a}_{i,t} | \mathbf{s}_{i,t} \right) \left(Q \left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t} \right) - V \left(\mathbf{s}_{i,t} \right) \right)$$

Overall, reinforcement learning is a very powerful technique to make an agent improve over time. One of its main drawbacks is that RL requires a lot of samples (i.e. millions of trajectory) to be able to learn successfully. Diplomacy has a long sequence of states with multiple orders per state. It is currently very difficult for RL algorithms to correctly determine what actions or sequence of actions cause the final game result (win/loss), which explains why a large number of samples are required to see an improvement in performance.

Chapter 3

The Diplomacy Game

3.1. Game Overview

Diplomacy is a 7-player game, where each player tries to conquer 18 out of the 34 supply centres on the map. The seven powers, namely Austria, England, France, Germany, Italy, Russia, and Turkey, start with either 3 or 4 units on the board. Units can either be armies or fleets. Armies can move on land and on coastal provinces, while fleets can move on water provinces and on coastal provinces.

The game starts in Spring 1901, and every power submits the orders for all their units at the same time. Orders become public and visible to all when they are adjudicated. For each year, there are a maximum of 5 possible phases, namely Spring Movement, Spring Retreats, Fall Movement, Fall Retreats, and Winter Adjustments.

To conquer a province, a unit needs to get support from adjacent units. An attack is successful only if the attack strength (unit attacking + number of units supporting the attack) is greater than the defense strength (unit defending + number of units supporting the defense). It is only possible to have a single unit on a province, so if an attack is successful, the dislodged unit is given a chance to retreat to an adjacent province during the retreat phase. Once per year, during the Winter phase, new units can be added or removed. If a power has more supply centers than units, it can build additional units on the provinces it had at the beginning of the game, but only if the power still controls them and they are unoccupied. If a power has more units than supply centers, it needs to remove some units

from the board to have the same number of units and supply centers.

Hold. A hold order (e.g. the army in Paris holds - *A PAR H*) can be submitted during movement phases to defend a province. It is also the default order that is submitted when a power does not submit any orders for a unit.

Move. A move order (e.g. the army in Paris moves to Burgundy - *A PAR - BUR*; the fleet in Mid-Atlantic moves to Spain (South Coast) - *F MAO - SPA/SC*; the army in London moves to Norway via convoy *A LON - NWY VIA*) can be submitted during movement phases to attack an adjacent province. Army units can move to adjacent land and coastal provinces, while fleet units can move to adjacent water provinces or to adjacent coastal provinces following a coast.

It is also possible for army units to move over several water provinces in a single turn by being convoyed by fleets. The convoyed army would submit an order that specifies its current province, the province it wants to move to, and a 'VIA' flag that indicates its intention to move via convoy. If there is a path where fleets issued the matching convoy orders, the army will be able to move to its destination.

Because there can only be one unit on a given province at a given time, a unit can bounce (i.e. the move order can fail) if two units move to the same province, or if a unit moves to a province but does not have enough attack strength to dislodge the unit on that province. Units are not allowed to swap provinces directly (i.e. *A PAR - BUR*, *A BUR - PAR* would fail), but can swap using a third province. It is common for units to bounce voluntarily, as it can be used to defend 3 provinces with 2 units. For instance, if France issues *A PAR - BUR*, *A MAR - BUR*, it can defend PAR, MAR and BUR with only 2 units.

Support (hold). A support hold order (e.g. the army in Paris supports the unit in Burgundy - *A PAR S A BUR*) can be submitted during movement phases to increase the

defense strength of a unit. A support hold order can be used to support any unit that is not moving. For instance, it is fairly common to create support chains where one unit issues a support move order, and a second unit issues a support hold on the first unit (i.e. *A GAS S A MAR - SPA*, *A PAR S A GAS*). The support hold order is invalid if the supported unit moves, or if another unit attacks the unit issuing the support order (i.e. the order is 'cut'). A unit can only support a unit in a province it could move into. For instance, *F TUS S A VEN* is invalid, because the fleet in Tuscany could not move to Venezia.

Support (move). A support move order (e.g. the army in Paris supports the army in Marseilles moving to Burgundy - *A PAR S A MAR - BUR*) can be submitted during movement phases to increase the attack strength of a unit. For the order to be valid, the unit issuing the order must be able to move to the destination of the move order, and the supported unit needs to issue a matching move order. For instance, *A PAR S A MAR - BUR* is valid, because the army in Paris is able to move to Burgundy. As for support hold orders, a support move order becomes invalid if the unit issuing the order is attacked (i.e. the support is 'cut').

Convoy. A convoy order (e.g. the fleet in North Sea convoys the army in London to Norway - *F NTH C A LON - NWY*) can be submitted during movement phases by fleets to allow an army to move over several water provinces. The convoy order must match exactly the 'move via' order, and the fleet must not be dislodged during the turn for the convoy order to remain valid.

Retreat. During retreat phases, a dislodged unit is allowed to retreat to an adjacent province by submitting a retreat order (e.g. the army in Paris retreats to Burgundy - *A PAR R BUR*). The retreat destination must be adjacent, unoccupied, not a standoff location (i.e. where a bounce occurred), and must not be the location where the attacker moved from. If two units retreat to the same province, or if no retreat locations are available, the dislodged

unit is disbanded and removed from the board.

Build Army / Fleet. During adjustment phases, if a power has more supply centers than units, it can build units to have the same number of units and supply centers by issuing build order (e.g. Build a fleet in Brest - *F BRE B*; Build an army in Rome - *A ROM B*). Units can only be built on one of the original supply center of that power if it still controls it and it is unoccupied.

Disband. During retreat and adjustment phases, it is possible to remove a unit from the board by issuing a disband order (e.g. the army in Paris disbands - *A PAR D*). A disband order is the default order if no orders are submitted for a dislodged unit during the retreat phase. For adjustment phases, if a power has more units than supply centers, extra units need to be disbanded, or they will be automatically disbanded according to a specific set of rules.

Waive. During adjustment phases, it is also possible for a power to skip its builds by issuing a waive order (i.e. *WAIVE*). A waive order is the default order in adjustment phases if the power has more supply centers than units. A power could waive builds to keep them for the future, or to signal it could be interested in drawing the game.

3.2. Communication in a No Press Game

In a No Press game, there are no messages, but players can still communicate with one another by using orders as signals [**37**]. In this section, I suggest some commonly used signals, mediated through orders, that communicate both strategic and tactical information. These signals are not part of the official rulebook, but are very commonly used in No Press games.

An order is considered valid if there is at least one scenario under which it could be executed successfully. It is important to understand how syntactically-correct but invalid

orders are treated. A *no-check* game is a variant where syntactically-correct but invalid orders can be submitted and viewed by other players (e.g. Russia: *A STP S A PAR - LON* and there are no units in Paris). Physical games are usually *no-check* games. In contrast, in a *check* game, only syntactically-correct valid orders are accepted, and invalid orders are ignored. I argue that, for reproducibility, all computer games should be *check* games and that, by default, any orders that cannot be successfully executed under any scenario should be discarded.

Declaring war. A player can effectively declare war by positioning its units in an offensive manner, by attacking another power, or by supporting an attack on another power.

Suggesting moves. A player can suggest a move by supporting or convoying the suggested move. For a support or convoy order where the destination is owned or occupied by a third power, the support might actually suggest the formation of an alliance against that third power. A support or convoy order used for signaling would be expected to be resubmitted at the next turn, but that is often not the case. For *no-check* games, a player could issue a support order for non-adjacent units or non-existent units to suggest making an alliance (e.g. Russia could do *A STP S A PAR - LON* to suggest France should attack England, even though PAR and STP are not adjacent to LON).

Proposing peace. Peace can be proposed by doing a support-hold on a foreign unit (e.g. Italy: *A VEN S A TRI* to propose peace to Austria). Even if the foreign unit moves and the support-hold is void, the foreign unit should be able to understand the underlying intention. For *no-check* games, peace can also be proposed by convoying a foreign unit to SWI (Switzerland is neutral and impassable).

Proposing draws. A draw can be proposed by convoying one unit of each power to SWI, by support-holding a unit of each power, or by simply deciding to waive builds.

Main article.

No Press Diplomacy: Modeling Multi-Agent Gameplay

par

Philip Paquette[1], Yuchen Lu[1], Steven Bocco[1], Max O. Smith[1], Satya Ortiz-Gagné[1], Jonathan K. Kummerfeld[1], Satinder Singh[1], Joelle Pineau[1], and Aaron Courville[1]

. The main contributions of each author are:

Philip Paquette negotiated an agreement with a third-party to access data, built the dataset and the game engine, did the majority of the work in conceptualizing and coding the supervised model, conceptualized and coded the reinforcement learning model, ran most of the experiments, and wrote the first version of the paper.

Yuchen Lu participated in weekly meetings, helped with the supervised model, ran some experiments, and helped writing the paper.

Steven Bocco helped in aggregating additional games in the dataset, built a web interface for the game engine, and provided his comments on the paper.

Max O. Smith participated in weekly meetings, and helped writing the paper.

Satya Ortiz-Gagné built a DAIDE-compatible module for the game engine, ran some experiments, and provided comments on the paper.

Jonathan K. Kummerfeld participated in weekly meetings, provided his insights on the performance of the agents, and helped writing the paper.

Other co-authors participated in weekly meetings and provided comments on the paper.

Résumé. Diplomacy est un jeu non-stochastique à sept joueurs, dans lequel les agents acquièrent des ressources à travers un mélange de travail d'équipe et de trahison. La dépendance à la confiance et à la coordination fait de Diplomacy la première référence multi-agents et non-coopérative pour résoudre des dilemmes sociaux séquentiels complexes dans un environnement riche. Dans cet ouvrage, nous nous concentrons sur la formation d'un agent qui apprend à jouer à la version No Press de Diplomacy, où il n'existe pas de canal de communication dédié entre les joueurs. Nous présentons *DipNet*, un modèle de politique basé sur un réseau de neurones pour No Press Diplomacy. Le modèle a été entraîné sur un nouvel ensemble de données de plus de 150 000 parties jouées par des humains. Notre modèle est formé par apprentissage supervisé (SL) à partir de trajectoires expertes, qui sont ensuite utilisées pour initialiser un agent d'apprentissage par renforcement (RL) qui joue contre des versions de lui-même. Les agents SL et RL démontrent des performances du niveau de l'état de l'art en battant les agents à base de règles.
Mots clés : Diplomacy, négociation, apprentissage supervisé, trahison, jeu, apprentissage par renforcement, théorie de l'esprit

Abstract. Diplomacy is a seven-player non-stochastic, non-cooperative game, where agents acquire resources through a mix of teamwork and betrayal. Reliance on trust and coordination makes Diplomacy the first non-cooperative multi-agent benchmark for complex sequential social dilemmas in a rich environment. In this work, we focus on training an agent that learns to play the No Press version of Diplomacy where there is no dedicated communication channel between players. We present *DipNet*, a neural-network-based policy model for No Press Diplomacy. The model was trained on a new dataset of more than 150,000 human games. Our model is trained by supervised learning (SL) from expert trajectories, which is then used to initialize a reinforcement learning (RL) agent trained through self-play. Both the SL and RL agents demonstrate state-of-the-art No Press performance by beating popular rule-based bots.
Keywords: Diplomacy, negotiation, supervised learning, betrayal, game, reinforcement learning, theory of mind

1. Introduction

Diplomacy is a seven-player game where players attempt to acquire a majority of supply centers across Europe. To acquire supply centers, players can coordinate their units with other players through dialogue or signaling. Coordination can be risky, because players can lie and even betray each other. Reliance on trust and negotiation makes Diplomacy the first non-cooperative multi-agent benchmark for complex sequential social dilemmas in a rich

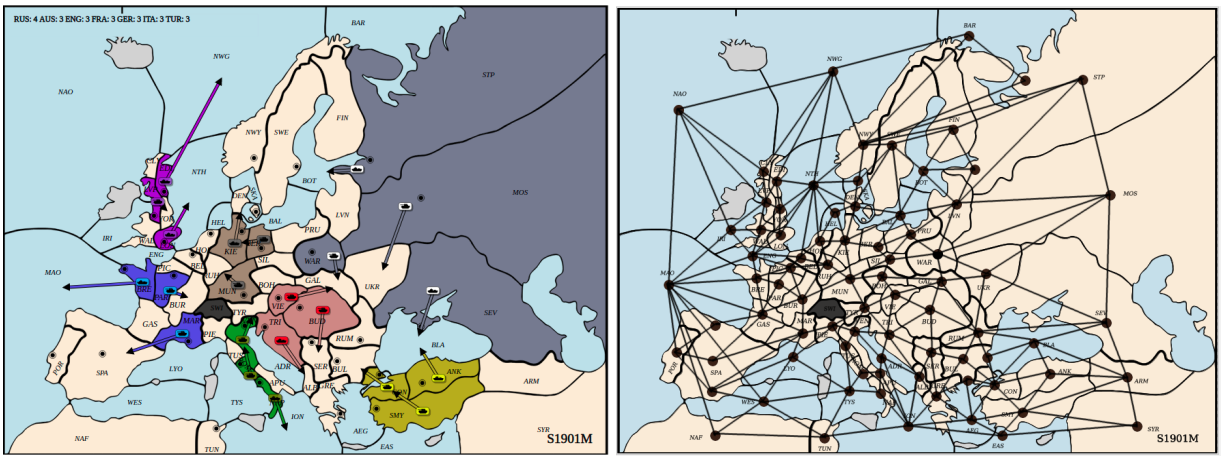
environment.

Sequential social dilemmas (SSD) are situations where one individual experiences conflict between self- and collective-interest over repeated interactions [**22**]. In Diplomacy, players are faced with a SSD in each phase of the game. Should I help another player? Do I betray them? Will I need their help later? The actions they choose will be visible to the other players and influence how other players interact with them later in the game. The outcomes of each interaction are non-stochastic. This characteristic sets Diplomacy apart from previous benchmarks where players could additionally rely on chance to win [**3, 4, 24, 28, 39**]. Instead, players must put their faith in other players and not in the game's mechanics (e.g., having a player role a critical hit).

Diplomacy is also one of the first SSD games to feature a rich environment. A single player may have up to 34 units, with each unit having an average of 26 possible actions. This astronomical action space makes planning and search intractable. Despite this, thinking at multiple time scales is an important aspect of Diplomacy. Agents need to be able to form a high-level long-term strategy (e.g. with whom to form alliances) and have a very short-term execution plan for their strategy (e.g. what units should I move in the next turn). Agents must also be able to adapt their plans, and beliefs about others (e.g. trustworthiness) depending on how the game unfolds.

In this work, we focus on training an agent that learns to play the No Press version of Diplomacy. The No Press version does not allow agents to communicate with each other using an explicit communication channel. Communication between agents still occurs through signalling in actions [**37, 3**]. This allows us to first focus on the key problem of having an agent that has learned the game mechanics, without introducing the additional complexity of learning natural language and learning complex interactions between agents.

We present *DipNet*, a fully end-to-end trained neural-network-based policy model for No Press Diplomacy. To train our architecture, we collect the first large scale dataset of Diplomacy, containing more than 150,000 games. We also develop a game engine that is



((a)) Orders submitted in S1901M **((b))** Map adjacencies.

Fig. 4.1. The Standard Diplomacy Map

compatible with DAIDE [**27**], a research framework developed by the Diplomacy research community, and that enables us to compare with previous rule-based state-of-the-art bots from the community [**38**]. Our agent is trained with supervised learning over the expert trajectories. Its parameters are then used to initialize a reinforcement learning agent trained through self-play.

In order to better evaluate the performance of agents, we run a tournament among different variants of the model as well as baselines and compute the TrueSkill score [**15**]. Our tournament shows that both our supervised learning (SL) and reinforcement learning (RL) agents consistently beat baseline rule-based agents. In order to further demonstrate the effect of architecture design, we perform an ablation study with different variants of the model, and find that our architecture has higher prediction accuracy for support orders even in longer sequences. This ability suggests that our model is able to achieve tactical coordination with multiple units. Finally we perform a coalition analysis by computing the ratio of cross-power support, which is one of the main methods for players to cooperate with each other. Our results suggest that our architecture is able to issue more effective cross-power orders.

2. No Press Diplomacy: Game Overview

Diplomacy is a game where seven European powers (Austria, England, France, Germany, Italy, Russia, and Turkey) are competing over supply centers in Europe at the beginning of the 20^{th} century. There are 34 supply centers in the game scattered across 75 provinces (board positions, including water). A power interacts with the game by issuing orders to army and fleet units. The game is split into years (starting in 1901) and each year has 5 phases: Spring Movement, Spring Retreat, Fall Movement, Fall Retreat, and Winter Adjustment.

Movements. There are 4 possible orders during a movement phase: Hold, Move, Support, and Convoy. A hold order is used by a unit to defend the province it is occupying. Hold is the default order for a unit if no orders are submitted. A move order is used by a unit to attack an adjacent province. Armies can move to any adjacent land or coastal province, while fleets can move to water or coastal provinces by following a coast.

Support orders can be given by any power to increase the attack strength of a moving unit or to increase the defensive strength of a unit holding, supporting, or convoying. Supporting a moving unit is only possible if the unit issuing the support order can reach the destination of the supported move (e.g. Marseille can support Paris moving to Burgundy, because an army in Marseille could move to Burgundy). If the supporting unit is attacked, its support is unsuccessful.

It is possible for an army unit to move over several water locations in one phase and attack another province by being convoyed by several fleets. A matching convoy order by the convoying fleets and a valid path of non-dislodged fleets is required for the convoy to be successful.

Retreats. If an attack is successful and there is a unit in the conquered province, the unit is dislodged and is given a chance to retreat. There are 2 possible orders during a retreat phase: Retreat and Disband. A retreat order is the equivalent of a move order, but only happens during the retreat phase. A unit can only retreat to a location that is 1) unoccupied, 2) adjacent, and 3) not a standoff location (i.e. left vacant because of a failed attack). A disband order indicates that the unit at the specified province should be removed from the

board. A dislodged unit is automatically disbanded if either there are no possible retreat locations, it fails to submit a retreat order during the retreat phase, or two units retreat to the same location.

Adjustments. The adjustment phase happens once every year. During that phase, supply centers change ownership if a unit from one power occupies a province with a supply center owned by another power. There are three possible orders during an adjustment phase: Build, Disband, and Waive. If a power has more units than supply centers, it needs to disband units. If a power has more supply centers than units, it can build additional units to match its number of supply centers. Units can only be built in a power's original supply centers (e.g. Berlin, Kiel, and Munich for Germany), and the power must still control the chosen province and it must be unoccupied. A power can also decide to waive builds, leaving them with fewer units than supply centers.

Communication in a No Press game. In a No Press game, even if there are no messages, players can communicate between one another by using orders as signals [**37**]. For example, a player can declare war by positioning their units in an offensive manner, they can suggest possible moves with support and convoy orders, propose alliances with support orders, propose a draw by convoying units to Switzerland, and so on. Sometimes even invalid orders can be used as communication, e.g., Russia could order their army in St. Petersburg to support England's army in Paris moving to London. This invalid order could communicate that France should attack England, even though Paris and St. Petersburg are not adjacent to London.

Variants. There are three important variants of the game: Press, Public Press, and No Press. In a Press game, players are allowed to communicate with one another privately. In a Public Press game, all messages are public announcements and can be seen by all players. In a No Press game, players are not allowed to send any messages. In all variants, orders are written privately and become public simultaneously, after adjudication. There are more than 100 maps available to play the game (ranging from 2 to 17 players), though the original Europe map is the most played, and as a result is the focus of

this work. The final important variation is *check*, where invalid orders are discarded, versus *no-check* where only valid orders are submitted. This distinction is important, because it determines the inclusion of a side-channel for communication through invalid orders.

Game end. The game ends when a power is able to reach a majority of the supply centers (18/34 on the standard map), or when players agree to a draw. When a power is in the lead, it is fairly common for other players to collaborate to prevent the leading player from making further progress and to force a draw.

Scoring system. Points in a diplomacy game are usually computed either with 1) a draw-based scoring system (points in a draw are shared equally among all survivors), or 2) a supply-center count scoring system (points in a draw are proportional to the number of supply centers). Players in a tournament are usually ranked with a modified Elo or TrueSkill system [**15**][**25**][**40**][**36**].

3. Previous Work

In recent years, there has been a definite trend toward the use of games of increasingly complexity as benchmarks for AI research including: Atari [**23**], Go [**33**][**34**], Capture the Flag [**17**], Poker [**4**][**24**], Starcraft [**39**], and DOTA [**28**]. However, most of these games do not focus on communication. The benchmark most similar to our No Press Diplomacy setting is Hanabi [**3**], a card game that involves both communication and action. However Hanabi is fully cooperative, whereas in Diplomacy, ad hoc coalitions form and degenerate dynamically throughout the evolution of the game. We believe this makes Diplomacy unique and deserving of special attention.

Previous work on Diplomacy has focused on building rule-based agents with substantial feature engineering. DipBlue [**11**] is a rule-based agent that can negotiate and reason about trust. It was developed for the DipGame platform [**10**], a DAIDE-compatible framework [**27**] that also introduced a language hierarchy. DBrane [**19**] is a search-based bot that uses branch-and-bound search, with state evaluation to truncate as appropriate. Another work, most similar to ours, uses self-play to learn a game strategy leveraging patterns of board

states [**32**]. Our work is the first attempt to use a data-driven method on a large-scale dataset.

Our work is also related to the learning-to-cooperate literature. In classical game theory, the Iterated Prisoner's Dilemma (IPD) has been the main focus for SSD, and a tit-for-tat strategy has been shown to be a highly effective strategy [**1**]. Recent work [**12**] has proposed an algorithm that takes into account the impact of one agent's policy on the update of the other agents. The resulting algorithm was able to achieve reciprocity and cooperation in both IPD and a more complex coin game with deep neural networks. There is also a line of work on solving social dilemmas with deep RL, which has shown that enhanced cooperation and meaningful communication can be promoted via causal inference [**18**], inequity aversions [**16**], and understanding consequences of intention [**30**]. However, most of this work has only been applied to simple settings. It is still an open question whether these methods could scale up to a complex domain like Diplomacy.

Our work is also related to behavioral game theory, which extends game theory to account for human cognitive biases and limitations [**6**]. Such behavior is observed in Diplomacy when players make non-optimal moves due to ill-conceived betrayals or personal vengeance against a perceived slight.

4. *DipNet*: A Generative Model of Unit Orders

4.1. Input Representation

Our model takes two inputs: current board state and previous phase orders. To represent the board state, we encode for each province: the type of province, whether there is a unit on that province, which power owns the unit, whether a unit can be built or removed in that province, the dislodged unit type and power, and who owns the supply center, if the province has one. If a fleet is on a coast (e.g. on the North Coast of Spain), we also record the unit information in the coast's parent province.

Previous orders are encoded in a way that helps infer which powers are allies and enemies. For instance, for the order 'A MAR S A PAR - BUR' (*Army in Marseille supports army in Paris moves to Burgundy*), we would encode: 1) 'Army' as the unit type, 2) the power

owning 'A MAR', 3) 'support' as the order type, 4) the power owning 'A PAR' (i.e. the friendly power), 5) the power, if any, having either a unit on BUR or owning the BUR supply center (i.e. the opponent power), 6) the owner of the BUR supply center, if it exists. Based on our empirical findings, orders from the last movement phase are enough to infer the current relationship between the powers. Our representation scheme is shown in Figure 4.2, with one vector per province.

4.2. Graph Convolution Network with FiLM

To take advantage of the adjacency information on the Diplomacy map, we propose to use a graph convolution-based encoder [**20**]. Suppose $x_{bo}^{l} \in \mathbb{R}^{81 \times d_{bo}^{l}}$ is the board state embedding produced by layer l and $x_{po}^{l} \in \mathbb{R}^{81 \times d_{po}^{l}}$ is the corresponding embedding of previous orders, where x_{bo}^{0}, x_{po}^{0} are the input representations described in Section 4.1.

We will now describe the process for encoding the board state; the process for the previous order embedding is the same. Suppose A is the normalized map adjacency matrix of 81×81. We first aggregate neighbor information by:

$$y_{bo}^{l} = BatchNorm(Ax_{bo}^{l}W_{bo} + b_{bo}) \tag{4.1}$$

where $W_{bo} \in \mathbb{R}^{d_{bo}^{l} \times d_{bo}^{l+1}}$, $b_{bo} \in \mathbb{R}^{d_{bo}^{l+1}}$, $y_{bo}^{l} \in \mathbb{R}^{81 \times d_{bo}^{l+1}}$ and $BatchNorm$ is operated on the last dimension. We perform conditional batch normalization using FiLM [**29, 31**], which has been shown to be an effective method of fusing multimodal information in many domains [**9**].

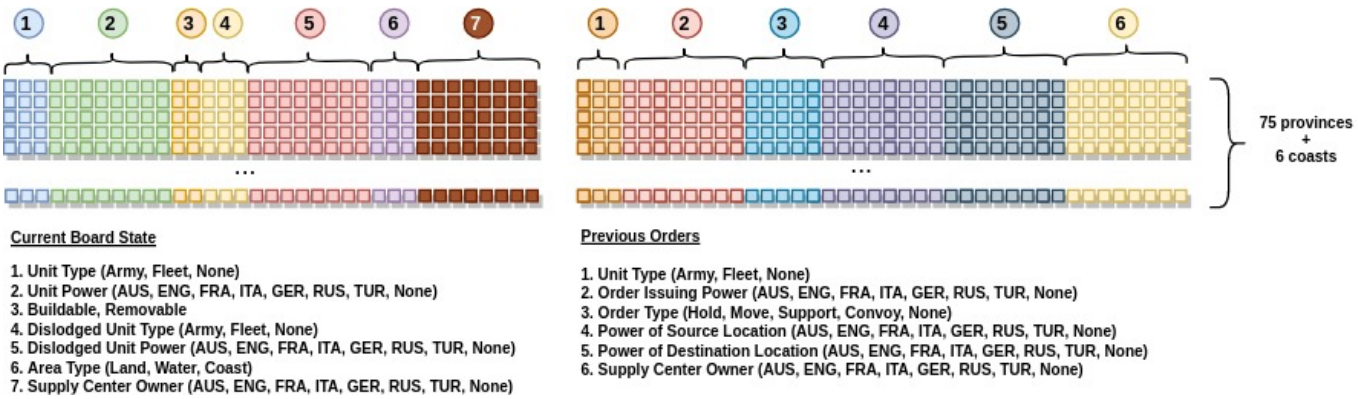


Fig. 4.2. Encoding of the Board State and Previous Orders.

Batch normalization is conditioned on the player's power p and the current season s (Spring, Fall, Winter).

$$\gamma_{bo}, \beta_{bo} = f^l_{bo}([p; s]) \qquad z^l_{bo} = y^l_{bo} \odot \gamma_{bo} + \beta_{bo} \tag{4.2}$$

where f_l is a linear transformation, γ,$\beta \in R^{d^{l+1}}$, and both addition and multiplication are broadcast across provinces. Finally we add a $ReLU$ and residual connections **[14]** where possible:

$$x^{l+1}_{bo} = \begin{cases} ReLU(z^l_{bo}) + x^l_{bo} & d^l = d^{l+1}_{bo} \\ ReLU(z^l_{bo}) & d^l_{bo} \neq d^{l+1}_{bo} \end{cases} \tag{4.3}$$

The board state and the previous orders are both encoded through L of these blocks, and there is no weight sharing. Concatenation is performed at the end, giving $h_{enc} = [x^L_{bo}, x^L_{po}]$ where h^i_{enc} is the final embedding of the province with index i. We choose $L = 16$ in our experiment.

4.3. Decoder

In order to achieve coordination between units, sequential decoding is required. However there is no natural sequential ordering. We hypothesize that orders are usually given to a cluster of nearby units, and therefore processing neighbouring units together would be effective. We used a top-left to bottom-right ordering based on topological sorting, aiming to prevent jumping across the map during decoding.

Suppose i^t is the index of the province requiring an order at time t, we use an LSTM to decode its order o^t by

$$h^t_{dec} = \text{LSTM}(h^{t-1}_{dec}, [h^{i^t}_{enc}; o^{t-1}]) \tag{4.4}$$

Then we apply a mask to only get valid possible orders for that location on the current board:

$$o^t = \text{MaskedSoftmax}(h^t_{dec}) \tag{4.5}$$

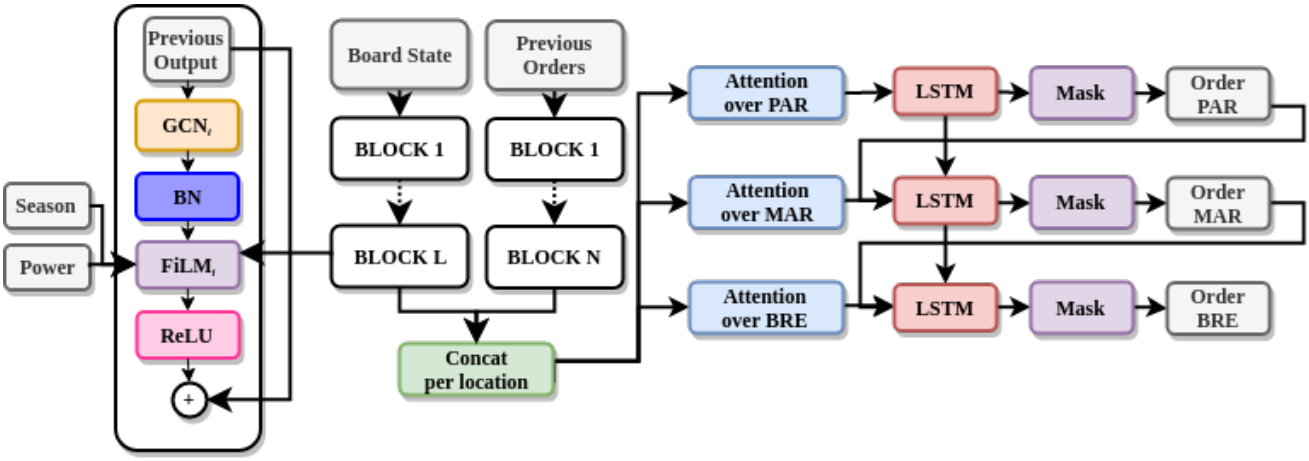


Fig. 4.3. DipNet Architecture

5. Datasets and Game Engine

Our dataset is generated by aggregating 156,468 anonymized human games. We also develop an open source game engine for this dataset to standardize its format and rule out invalid orders. The dataset contains 33,279 No Press games, 1,290 Public Press games, 105,266 Press games (messages are not included), and 16,633 games not played on the standard map. We are going to release the dataset along with the game engine[1]. Detailed dataset statistics are shown in Table 4.1.

The game engine is also integrated with the Diplomacy Artificial Intelligence Development Environment (DAIDE) [**27**], an AI framework from the Diplomacy community. This enables us to compare with several state-of-the-art rule-based bots [**38, 11**] that have been developed

[1]Researchers can request access to the dataset by contacting the first author. An executive summary describing the research purpose and execution of a confidentiality agreement are required.

on DAIDE. DAIDE also has a progression of 14 symbolic language levels (from 0 to 130) for negotiation and communication, which could be potentially useful for research on Press Diplomacy. Each level defines what tokens are allowed to be exchanged by agents. For instance, a No Press bot would be considered level 0, while a level 20 bot can propose peace, alliances, and orders.

6. Experiments

6.1. Supervised Learning

We first present our supervised learning results. Our test set is composed of the last 5% of games sorted by game id in alphabetical order. To measure the impact of each model component, we ran an ablation study. The results are presented in Table 4.2. We evaluate the model with both greedy decoding and teacher forcing. We measure the accuracy of each unit-order (e.g. 'A PAR - BUR'), and the accuracy of the complete set of orders for a power (e.g. 'A PAR - BUR', 'F BRE - MAO'). We find that our untrained model with a masked decoder performs better than the random model, which suggests the effectiveness of masking out invalid orders. We observe a small drop in performance when we only provide the board state. We also observe a performance drop when we use the average embedding over all

Tab. 4.1. Dataset statistics

				Survival rate for opponents						
	Win%	Draw%	Defeated%	AUS	ENG	FRA	GER	ITA	RUS	TUR
Austria	4.3%	33.4%	48.1%	100%	79%	62%	55%	40%	29%	15%
England	4.6%	43.7%	29.1%	47%	100%	30%	16%	49%	33%	80%
France	6.1%	43.8%	25.7%	40%	26%	100%	22%	45%	59%	77%
Germany	5.3%	35.9%	40.4%	44%	26%	39%	100%	61%	27%	80%
Italy	3.6%	36.5%	40.2%	15%	65%	56%	61%	100%	56%	25%
Russia	6.6%	35.2%	39.8%	25%	52%	77%	38%	63%	100%	42%
Turkey	7.2%	43.1%	26.0%	9%	78%	71%	56%	23%	31%	100%
Total	39.9%	60.1%		37%	59%	65%	49%	51%	50%	64%

Tab. 4.2. Evaluation of supervised models: Predicting human orders.

Model	Accuracy per unit-order		Accuracy for all orders	
	Teacher forcing	Greedy	Teacher forcing	Greedy
DipNet	**61.3**%	**47.5**%	**23.5**%	**23.5**%
Untrained	6.6%	6.4%	4.2%	4.2%
Without FiLM	60.7%	47.0%	22.9%	22.9%
Masked Decoder (No Board)	47.8%	26.5%	14.7%	14.7%
Board State Only	60.3%	45.6%	22.9%	23.0%
Average Embedding	59.9%	46.2%	23.2%	23.2%

locations as input to the LSTM decoder (rather than using attention based on the location the current order is being generated for).

To further demonstrate the difference between these variants we focus on the model's ability to predict support orders, which are a crucial element for successful unit coordination. Table 4.3 shows accuracy on this order type, separated based on the position of the unit in the prediction sequence. We can see that although the performance of different variants of the model are close to each other when predicting support for the first unit, the difference is larger when predicting support for the 16^{th} unit. This indicates that our architecture helps DipNet maintain tactical coordination across multiple units.

Tab. 4.3. Comparison of the models' ability to predict support orders with greedy decoding.

	Support Accuracy	
	1^{st} location	16^{th} location
DipNet	**40.3%**	**32.2%**
Board State Only	38.5%	25.9%
Without FiLM	40.0%	30.3%
Average Embedding	39.1%	27.9%

6.2. Reinforcement Learning and Self-play

We train *DipNet* with self-play (same model for all powers, with shared updates) using an A2C architecture [**23**] with n-step (n=15) returns for approximately 20,000 updates (approx. 1 million steps). As a reward function, we use the average of (1) a local reward function (+1/-1 when a supply center is gained or lost (updated every phase and not just in Winter)), and (2) a terminal reward function (for a solo victory, the winner gets 34 points; for a draw, the 34 points are divided proportionally to the number of supply centers). The policy is pre-trained using DipNet SL described above. We also used a value function pre-trained on human games.

The opponents we have used to evaluate our agents were: **(1) Random.** This agent selects an action per unit uniformly at random from the list of valid orders. **(2) GreedyBot.** This agent greedily tries to conquer neighbouring supply centers and is not able to support any attacks. **(3) Dumbbot [26].** This rule-based bot computes a value for each province, ranks orders using computed province values and uses rules to maintain coordination. **(4) Albert Level 0 [38].** Albert is the current state-of-the-art agent. It evaluates the

Tab. 4.4. Diplomacy agents comparison when played against each other, with one agent controlling one power and the other six powers controlled by copies of the other agent.

Agent A (1x)	Agent B (6x)	TrueSkill A-B	% Win	% Most SC	% Survived	% Defeated	# Games
SL DipNet	Random	28.1 - 19.7	100.0%	0.0%	0.0%	0.0%	1,000
SL DipNet	GreedyBot	28.1 - 20.9	97.8%	1.2%	1.0%	0.0%	1,000
SL DipNet	Dumbbot	28.1 - 19.2	74.8%	9.2%	15.4%	0.6%	950
SL DipNet	Albert 6.0	28.1 - 24.5	28.9%	5.3%	42.8%	23.1%	208
SL DipNet	RL DipNet	28.1 - 27.4	6.2%	0.3%	41.4%	52.1%	1,000
Random	SL DipNet	19.7 - 28.1	0.0%	0.0%	4.4%	95.6%	1,000
GreedyBot	SL DipNet	20.9 - 28.1	0.0%	0.0%	8.5%	91.5%	1,000
Dumbbot	SL DipNet	19.2 - 28.1	0.0%	0.1%	5.0%	95.0%	950
Albert 6.0	SL DipNet	24.5 - 28.1	5.8%	0.4%	12.6%	81.3%	278
RL DipNet	SL DipNet	27.4 - 28.1	14.0%	3.5%	42.9%	39.6%	1,000

probability of success of orders, and builds alliances and trust between powers, even without messages. To evaluate performance, we run a tournament and compute TrueSkill scores for these models **[15]**. Games in the tournament are structured with one power controlled by one agent and the other six controlled by copies of another agent. We report the results of the first agent in Table 4.4. From the TrueSkill score we can see both the SL (28.1) and RL (27.4) versions of DipNet consistently beat the baseline models as well as Albert (24.5), the previous state-of-art bot. Although there is no significant difference in TrueSkill between SL and RL, the performance of RL vs 6 SL is better than SL vs 6 RL with an increasing win rate.

6.3. Coalition Analysis

In the No Press games, cross-power support is the major method for players to signal and coordinate with each other for mutual benefit. In light of this, we propose a coalition analysis method to further understand agents' behavior. We define a cross-power support (X-support) as being when a power supports a foreign power, and we define an effective cross-power support as being a cross-power order support without which the supported attack or defense would fail:

$$\textit{X-support-ratio} = \frac{\#\text{X-support}}{\#\text{support}}, \qquad \textit{Eff-X-support-ratio} = \frac{\#\text{Effective X-support}}{\#\text{X-support}}$$

The *X-support-ratio* reflects how frequently the support order is used for cooperation/communication, while the *Eff-X-support-ratio* reflects the efficiency or utility of cooperation. We launch 1000 games with our model variants for all powers and compute this ratio for each one. Our results are shown in Table 4.5.

For human games, across different game variants, there is only minor variations in the *X-support-ratio*, but the *Eff-X-support-ratio* varies substantially. This shows that when people are allowed to communicate, their effectiveness in cooperation increases, which is consistent with previous results that cheap talk promotes cooperation for agents with

aligned interests [**7, 8**]. In terms of agent variants, although RL and SL models show similar TrueSkill scores, their behavior is very different. RL agents seem to be less effective at cooperation but do have more frequent cross-power support. This decrease in effective cooperation is also consistent with past observations that naive policy gradient methods fail to learn cooperative strategies in a non-cooperative setting such as the iterated prisoner dilemma [**13**]. Ablations of the SL model have a similar *X-support-ratio*, but suffer from a loss in *Eff-X-support-ratio*. This further suggests that our DipNet architecture can help agents cooperate more effectively. The Masked Decoder has a very high *X-support-ratio*, suggesting that the marginal distribution of support is highest among agent games, however, it suffers from an inability to effectively cooperate (i.e. very small *Eff-X-support-ratio*). This is also expected since the Masked Decoder has no board information to understand the effect of supports.

Tab. 4.5. Coalition formation: Diplomacy agents comparison

		X-support-ratio	*Eff-X-support-ratio*
Human Game	No Press	14.7%	7.7%
	Public Press	11.8%	12.1%
	Press	14.4%	23.6%
Agents Games	RL DipNet	9.1%	5.3%
	SL DipNet	7.4%	10.2%
	Board State Only	7.3%	7.5%
	Without FiLM	6.7%	7.9%
	Masked Decoder (No Board)	12.1%	0.62%

7. Conclusion

In this work, we present *DipNet*, a fully end-to-end policy for the strategy board game No Press Diplomacy. We collect a large dataset of human games to evaluate our architecture. We train our agent with both supervised learning and reinforcement learning self-play. Our tournament results suggest that DipNet is able to beat state-of-the-art rule-based bots in

the No Press setting. Our ablation study and coalition analysis demonstrate that DipNet can effectively coordinate units and cooperate with other players. We propose Diplomacy as a new multi-agent benchmark for dynamic cooperation emergence in a rich environment. Probably the most interesting result to emerge from our analysis is the difference between the SL agent (trained on human data) and the RL agent (trained with self-play). Our coalition analysis suggests that the supervised agent was able to learn to coordinate support orders while this behaviour appears to deteriorate during self-play training. We believe that the most exciting path for future research for Diplomacy playing agents is in the exploration of methods such as LOLA **[13]** that are better able to discover collaborative strategies among self-interested agents.

Chapter 4

Conclusion

In this thesis, I introduced a supervised and a reinforcement learning model that can play the board game No Press Diplomacy. The model was trained using a new dataset of more than 150,000 human games and a newly-developed game engine. The proposed *DipNet* model is able to beat state-of-the-art rule-based bots. Moreover, the model demonstrates an ability to coordinate units and cooperate with other players.

DipNet uses a combination of graph convolutions, conditional batch normalization, an attention mechanism, and a masked decoder. The model has the best performance when it is trained to replicate the orders of all the players in both Press and No Press games.

Even though the model is able to win against rule-based agents, a lot of research will be required to reach super-human performance. Some interesting research directions might include 1) having agents create a language between themselves to beat agents that cannot communicate, 2) having agents learn strategies that can be applied to different maps, 3) having agents negotiate with humans in natural language, or 4) having agents detect when other players are being deceitful.

Bibliography

[1] Robert Axelrod and William D Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *arXiv preprint arXiv:1902.00506*, 2019.

[4] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[5] Jay Burmeister and Janet Wiles. The challenge of go as a domain for ai research: a comparison between go and chess. In *Proceedings of Third Australian and New Zealand Conference on Intelligent Information Systems. ANZIIS-95*, pages 181–186. IEEE, 1995.

[6] Colin F Camerer, Teck-Hua Ho, and Juin Kuan Chong. Behavioural game theory: thinking, learning and teaching. In *Advances in Understanding Strategic Behaviour*, pages 120–180. Springer, 2004.

[7] Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. Emergent communication through negotiation. *arXiv preprint arXiv:1804.03980*, 2018.

[8] Vincent P Crawford and Joel Sobel. Strategic information transmission. *Econometrica: Journal of the Econometric Society*, pages 1431–1451, 1982.

[9] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 3(7):e11, 2018.

[10] Angela Fabregues, David Navarro, Alejandro Serrano, and Carles Sierra. Dipgame: A testbed for multiagent systems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1619–1620. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[11] André Ferreira, Henrique Lopes Cardoso, and Luis Paulo Reis. Dipblue: A diplomacy agent with strategic and trust reasoning. In *ICAART 2015-7th International Conference on Agents and Artificial Intelligence, Proceedings*, 2015.

[12] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[13] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{TM}}$: a bayesian skill rating system. In *Advances in neural information processing systems*, pages 569–576, 2007.

[16] Edward Hughes, Joel Z Leibo, Matthew Phillips, Karl Tuyls, Edgar Dueñez-Guzman, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin McKee, Raphael Koster, et al. Inequity aversion improves cooperation in intertemporal social dilemmas. In *Advances in Neural Information Processing Systems*, pages 3326–3336, 2018.

[17] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

[18] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando de Freitas. Intrinsic social motivation via causal influence in multi-agent rl. *arXiv preprint arXiv:1810.08647*, 2018.

[19] Dave Jonge and Jordi González Sabaté. Negotiations over large agreement spaces, 2015.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Albert Lamorisse. Risk, the world conquest game. Parker Brothers, 1993.

[22] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[24] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[25] Tony Nichols. A player rating system for diplomacy. `http://www.stabbeurfou.org/docs/articles/en/DP_S1998R_Diplomacys_New_Rating_System.html`, 1998. Accessed: 2019-05-01.

[26] David Norman. Daide - clients. `http://www.daide.org.uk/clients.html`, 2013. Accessed: 2019-05-01.

[27] David Norman. Daide - diplomacy artificial intelligence development environment. `http://www.daide.org.uk/`, 2013. Accessed: 2019-05-01.

[28] OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

[29] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[30] Alexander Peysakhovich and Adam Lerer. Consequentialist conditional cooperation in social dilemmas with imperfect information. *arXiv preprint arXiv:1710.06975*, 2017.

[31] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[32] Ari Shapiro, Gil Fuchs, and Robert Levinson. Learning a game strategy using pattern-weights and self-play. In *International Conference on Computers and Games*, pages 42–60. Springer, 2002.

[33] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[34] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[36] super dipsy. Site scoring system. `https://www.playdiplomacy.com/forum/viewtopic.php?f=565&t=34913`, 2013. Accessed: 2019-05-01.

[37] Simon Szykman. Dp w1995a: Communication in no-press diplomacy. `http://uk.diplom.org/pouch/Zine/W1995A/Szykman/Syntax.html`, 1995. Accessed: 2019-05-01.

[38] Jason van Hal. Diplomacy ai - albert. `https://sites.google.com/site/diplomacyai/`, 2013. Accessed: 2019-05-01.

[39] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[40] WebDiplomacy. Ghost-ratings explained. `https://sites.google.com/view/webdipinfo/ghost-ratings/ghost-ratings-explained`, 2019. Accessed: 2019-05-01.