Diplomacy AI in Variants

Maxim Popov, Pratik Maitra

8th May, 2024

Abstract

Our research focuses on application of small scale AI models for playing the game of diplomacy. In our work we demonstrate how to apply the latest principles of parameter efficient fine tuning to build a tiny diplomacy AI agent. We also list the observations and the other nuances related to our work.

1 Introduction

1.1 Artificial Intelligence in Games

Artificial Intelligence has been incredibly successful in the realm of strategy games over the last several decades. AI has surpassed top human level performance in a variety of these games ranging from Deep Blue's victory against world champion Gary Kasparov in chess in the late 1990's to more recent super-human performance in positional games such as go and video games such as Starcraft. Recently, incredibly strong, although not yet super-human, AI performance has been brought to the realm of Diplomacy, a complex mixed cooperative-competitive game.

Taking a step back, the development of artificial intelligence systems in games is important to acquire skills and techniques to apply to real world scenarios. Even the basics of learning and winning in complex competitive environments is very applicable in the real world where just about any field can be viewed as some competition to win with some environment to overcome. Some games, however, provide a further challenge with a greater promise of reward. Diplomacy is one such game. With a requirement for cooperation in a shifting dynamic alliance structure dictated by player will alone and effective communication necessary for accomplishment thereof all in order to surpass every other player including former allies, an AI system that can master the game of Diplomacy can learn to cooperate, communicate, and win all at once while also displaying the effectiveness of minimizing lies in competitive communication environments. This can be applied in any human-computer interaction and lends itself directly to incredibly delicate real world situations such as real diplomacy between real states which decides the fate of all of the people on Earth.[5]

1.2 Motivation for our work

Current state of the art diplomacy AIs have far surpassed human tactics in 2-player Diplomacy variants, reached human level in non-press communication and cooperation in the non-press gunboat classic variant, and done likewise even full press no limitations blitz 5-minute turns against top human players. While there is further room to grow in defeating humans in longer deadline games and surpassing top-level performance in any full-press games, there is also room for growth in developing artificial intelligence systems to attain such performance in novel variant environments following many of the same basic rules along the lines of competition, communication, and cooperation. Toward that end, in this work we seek to explore AI systems in these variant environments beginning with no rule changes beyond a novel map layout on the board, an extension to which will pave the way for further work toward broadening capabilities and employing existing classical game data to other variants.

1.3 Diplomacy Rules

In this section we provide an overview of the rules of Diplomacy for a better understanding of our work. Diplomacy is played on a map of early 20th century Europe where each of seven players each controlling one European great power attempt to control the majority of the 34 supply centers on the board. Each beginning with a pre-determined set of either 3 or 4 centers, they must defeat all opponents and become the sole dominating power in Europe.

The way that the players interact with the board is that they each control one unit per supply center they own. These units have a small set of orders they can accomplish consisting of holding in place, moving into a neighboring province, supporting another unit's move to a neighboring province, supporting another unit's hold in place, and, if a fleet, convoying an army from one land space to another.

All actions are adjudicated simultaneously. Thus, in each year denomination of the game, all units shall move simultaneously in the Spring, do likewise in the Fall, and then units will be placed or taken off of the board in the Winter with respect to a given player's supply center count before repeating in the following game year. The major addition to these rules is that during the course of each turn, players have a predetermined amount of time to communicate with one another in cheap talk, making any agreements they wish but not being bound to any of them.

It is clear that the more units one controls the more they can accomplish and grow. Therefore, multiple players working together can dispatch another player and split the fallen supply centers amongst themselves. Any set of players cooperating have an overpowering advantage against non-cooperating players. However, since only one player can win, cooperation cannot last forever and given the nature of cheap talk, alliances can shift on any turn.

Thus, in this environment, an AI system must be able to provide orders to its units in such a way as to maximize those units' utility in acquiring new supply centers and holding onto old ones. In this sense it is a similar game to chess except for the added complication of a much larger state space given each of typically 34 units can make one of a dozen or two moves on a given turn. In addition to this, there are 6 other players rather than one who will choose only a subset of the other players to fight against and another subset to attempt to work together with. More difficult than that, even, is the importance of communicating with other players to build trust and work together while coming out on top of that relationship in the end. This all creates a complicated web to manage for any AI system.

A variant of this classic version of the game can be anything that changes any rule. In our case, we seeked to explore solely changing the map layout into a different set of province adjacencies and supply center layout, for example playing on a map of North America instead of Europe with some potentially different number of players but all mechanical aspects of the game remaining constant. Given that AI systems have been developed to perform well on the classic map, they can do likewise here if the abilities and learning can be ported from data which is plentiful on the classic map and nearly or entirely non-existent on variants.

2 Related Works

The last couple of years has seen the majority of the advancement in Diplomacy artificial intelligence systems. Initial research focused solely on the no-press gunboat variant in which the challenge was to optimize unit orders without worrying about written communication. It is important to note that communication is still involved and important as orders, sometimes illegal ones (resulting in a hold but still revealed to other players), can still be used to signal alliance. The year 2020 saw the first human level (but not yet superhuman-level) performance in this no-press variant [3]. These models notably trained on a large store of past human games on the Diplomacy website webdiplomacy.net.

Further research into training a model from scratch, without using the available data, found that superhuman performance was achieved in a popular 2-player variant. However, when the model was trained on the classic map without press, it performed significantly worse than existing models despite its strong tactical abilities, suggesting that there are multiple Nash equilibria and that it did not find the right one. [1]

More work was completed to further improve upon classic gunboat employing reinforcement learning but also using the existing datastores. Another technique used was to model human play to improve coordination by penalizing moves a human would not typically make, thus limiting them unless they were incredibly powerful [2]. These works achieved top-level human performance in tournament settings, but still have yet to surpass them.

The most recent work has delved into the full game without a ban on cheap talk press. The tactical model was combined alongside a large language model to both talk to players and submit powerful moves. The resulting Cicero model has achieved top-level human performance in 5-minute games. This agent learned that lying should be minimized to maximize trust and effectiveness in both making and exploiting alliances. [5]

On the technical side of things our work is based on two relatively new approaches for fine-tuning of Large Language Models in recent years viz Parameter Efficient Fine Tuning[6] and Low Rank Adaptation of Large Language Models[7]. We go into great detail about how we implemeted these two paradigms while training our model.

3 Methodology

3.1 Initial Objectives

Our initial goal was to extend present state of the art techniques to arbitrary map variants while maintaining all other rules and comparing the results. The idea, as discussed earlier, was twofold. First, the ability to extend the environment is important as oftentimes environments shift and oftentimes similar environments can learn from past successes in other environments. Second, we could explore how the large store of data could be used with the same rules but different adjacencies.

As such, we planned to parse arbitrary maps of which many exist, train a no-press model from scratch, a no-press model which attempted to not diverge too significantly from human play, and a full-press Cicero-like model to play on both some specific variant and on arbitrary variants.

The expected challenges were a mix of old existing challenges and new ones. As per old challenges, the state space of Diplomacy is over 10^{20} . It is not feasible to compute a search as one might in chess. We would also have lesser access to training hardware. New challenges would include determining how to exploit the large datastore in a different but similar environment and how to train a model to work well on any arbitrary variant as opposed to one particular one.

Unfortunately, as will be discussed further on, we ran into insurmountable challenges and were forced to pivot our objectives to something much less powerful due to an inability to build upon state of the art solutions. However, we shall still describe the arbitrary map parser that was developed.

3.2 Map Parser

To parse an arbitrary map, we must obtain the provinces, their adjacencies, their type (land or water), their owner (player or neutral), their center status (whether or not a supply center exists on that province), and their unit status (which unit if any exists on that province). As such, the input is simply three files in the format that is common in Diplomacy map design: a background province view file, a center file, and a unit file.

To parse the images, we first find the province locations. To do this, we label the spaces inside the given border color with unique IDs using a flood fill algorithm. We then expand these spaces into the border so that the IDs border each other directly. From there on we find all adjacent IDs and create an adjacency matrix. To determine province owner and type, we parse the colors on the original image that define such. Center and unit statuses are found by means of locating the province that the center and unit images fall under.

The resulting data: province, center, and unit information, is ready to be sent in whatever format required by state of the art solutions to first attempt a minimally-modified version thereof. Unfortunately at this stage we could not move forward with our original plans due to difficulties in using these systems as described below.

3.3 Model

We decided to train a transformer based LLM model from scratch as we faced a few issues with CICERO[10]. CICERO is based on the R2C2 architecture, which is a 2.7B parameter transformer-based encoder-decoder model. The first issue we had with CICERO was the lack of support and maintenance of the CICERO github repository. The repository had been updated about two years ago and many of the python libraries were either incompatible or broken. Another problem with CICERO was that while being open source it required a lot of computational resources to operate. One estimate put the computational power of 56 GPUs for it to run. Hence it would be infeasible for us to customize CICERO for our use case or even fine tune it.

Therefore we decided to fine-tune our own LLM based diplomacy agent and trained our own small scale LLM from scratch on a synthetic dataset. We believe that a small lightweight model with correct prompting can mimic the planning and reinforcement learning stratagem applied by CICERO.

The model we chose for training is the "TinyLlama-1.1B-Chat-v1.0"[9]. It is a 1.1B Llama model pre-trained on 3 trillion tokens. The reason for selecting an open source small parameter based LLM variant is that it could be trained on our custom dataset and it‘s training would be relatively computationally less expensive.

The key paradigm behind the training of our model was based on the idea of PEFT(Parameter Efficient Fine Tuning) and LoRA(Low Rank Adaptation of Large Models) as reference in the related works section.

PEFT fine-tunes only a small number of (extra) model parameters while freezing most parameters of the pretrained LLMs, thereby greatly decreasing the computational and storage costs. PEFT methods include LoRA, Prefix Tuning, P-tuning and Prompt tuning. PEFT approaches enables one to get performance comparable to full fine-tuning while only having a small number of trainable parameters

LoRA is a lightweight training technique which reduces the number of trainable parameters. It is a form of PEFT and works by inserting a smaller number of new weights into the model and only these are trained. This makes training with LoRA much faster, memory-efficient, and produces smaller model weights (a few hundred MBs), which are easier to store and share. LoRA can also be combined with other training techniques to speedup training. LoRA is currently used for fine tuning a lot of LLMs at greatly reduced computational costs.

We implemented LoRA(PEFT) using the PEFT library which is already integrated with Transformers and accelerate library.

3.4 Training

We started by creating the synthetic dataset for training the tiny llama model. The dataset consisted of 100 completed diplomacy games and along with instructions from CICERO‘s redacted games dataset. The CICERO team had used RL for training it‘s model and their github repository included a collection of diplomacy games and instructions. Since the number of games in the CICERO dataset was insufficient to train our dataset, we generated a dataset of 100 completed diplomacy games and extracted the diplomacy commands for each nation. Our combined training dataset amounted to about 20,000 rows of diplomacy commands.

As mentioned previously, the training of the tiny llama model was based on two key paradigms, viz LoRA and PEFT.

For fine tuning the model we used a prompt that reinforces the LLM chatbot agent of tiny llama. Our training prompt was ”You are an AI agent playing Diplomacy. Please give the orders for your turn.” We ran the training exercise over the entire dataset.

The hardware used for training was a personal RTX4090 along with the free GPU offered by Google Colab Pro. The hyperparameters for training our model is provided in the appendix C section of the paper.

4 Experiments

The experimental aspect of our task was to perform an evaluation of the tiny llama model in a diplomacy game. To give a human-like aspect to our model, we sometimes changed the temperature setting of our prompt. The temperature aspect of prompting adds a bit of variance to the model output. Our evaluation goal is to observe how viable our model is at playing Diplomacy and the quality of the commands it gives when compared to other LLMs like Chatgpt etc. The evaluation metric was the accuracy of the prompt response.

Due to the lack of any automated tool for judging the quality as well as accuracy of the commands, the evaluation was done manually. The manually annotated evaluation results are provided in the repository.

5 Results

We managed to play an entire game using the tiny llama bot. There were several instances where we had to repeat the prompt for a valid order or apply a random order but the tiny llama model does have the capability to give valid moves for diplomacy. We got correct commands for the majority of the prompts(with low temperature settings) but the model sometimes produced nonsensical or wrong commands. The overall accuracy for the model was roughly 60% of the prompts for the one with 0.1 temperature value setting and 30% for the one with 1 temperature value settings. When compared to ChatGPT it seems that ChatGPT can understand the prompts better than our model but we have to consider the difference in the number of parameters between the two LLMs. The average response time of the model(hardware dependent) varied from 0.76 s to 0.83 s .

6 Conclusion

We believe our project showcased how a relatively small model can be finetuned for a complex task. Our model is well suited for running on edge devices and most significantly can provide a viable alternative to larger more complex LLMs.

We summarize our findings in two key insights as provided below:

- Our experiment showed that it is indeed possible to leverage PEFT techniques like LoRA to build workable LLM agents which are based on tiny LLMs.

- Our preliminary analysis reveals that larger parameter LLMs perform better than ones with fewer parameters and the volume of data may also play a large role in final performance.

7 Future Work

Finally we also would like to apply the proof of concept technique of LoRA to other more challenging tasks. Some ideas for future work can include extending the methodology to other newly released models. Another avenue for future work is training a similar model for other games like chess.

8 Acknowledgement

We would like to thank the Professor and the teaching assistant for providing invaluable feedback for our project.

REFERENCES

[1] Bakhtin, Anton, et al. “Mastering the game of no-press Diplomacy via human-regularized reinforcement learning and planning.” *arXiv preprint arXiv:2210.05492*, 2022.

[2] Bakhtin, Anton, et al. “No-press diplomacy from scratch.” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18063–18074, 2021.

[3] Gray, Jonathan, et al. “Human-level performance in no-press diplomacy via equilibrium search.” *arXiv preprint arXiv:2010.02923*, 2020.

[4] Jacob, Athul Paul, et al. “Modeling strong and human-like gameplay with KL-regularized search.” *International Conference on Machine Learning*. PMLR, 2022.

[5] Meta Fundamental AI Research Diplomacy Team (FAIR), et al. “Human-level play in the game of Diplomacy by combining language models with strategic reasoning.” *Science*, vol. 378, no. 6624, pp. 1067–1074, 2022.

[6] Ding, N., Qin, Y., Yang, G., et al. “Parameter-efficient fine-tuning of large-scale pre-trained language models.” *Nat Mach Intell*, vol. 5, pp. 220–235, 2023. https://doi.org/10.1038/s42256-023-00626-4

[7] Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. ”Lora: Low-rank adaptation of large language models.” arXiv preprint arXiv:2106.09685 (2021). https://arxiv.org/abs/2106.09685

[8] https://github.com/PratikMaitra/COMS_672_Project

[9] (https://github.com/jzhang38/TinyLlama)

[10] https://github.com/facebookresearch/diplomacy_cicero

[11] https://github.com/maximpopov11/DiplomacyGM/tree/main.

[12] https://github.com/mukobi/welfare-diplomacy

Appendix A Code Link

The Github for Map Parser at [11] `https://github.com/maximpopov11/DiplomacyGM/tree/main`.

The Map Parser was built to parse arbitrary Diplomacy maps in order to then feed them into existing models. Unfortunately, as explained above, we could not use those models and as such this parser was not used. We include it to present progress along that track prior to pivoting away.

The project can be found at [8] `https://github.com/PratikMaitra/COMS_672_Project`.

The map parser and the UI can also be accessed using the links [11] and [12].

It has it's own map parser and one can substitute it with our parser.

Appendix B README

The above github repository contains the code for training and performing inference for the model.

The simplest way to run the final model and get diplomacy commands is to run the TinyllamaDiplo.ipynb Jupyter Notebook file in a system with GPU support/Google collab with the free GPU. This should enable one to run the inference code for our model.

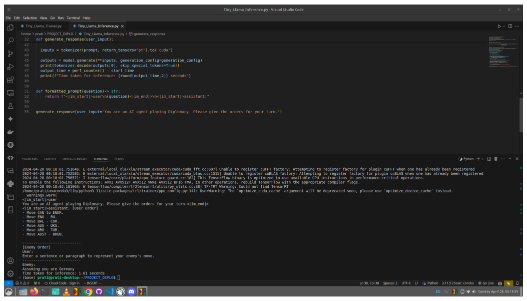
The OS environment for the entire experiment was Ubuntu 23.10. It is preferred to either use the Ubuntu OS or perform the inference on Google Colab.

The dataset for training is provided in the github repository under the ”dataset” folder.

Appendix C Tables and Figures

Parameter	**Value**
Output Directory	`output_model`
Per Device Train Batch Size	2
Gradient Accumulation Steps	4
Optimizer	`paged_adamw_32bit`
Learning Rate	2×10^{-4}
LR Scheduler Type	`cosine`
Save Strategy	`epoch`
Logging Steps	10
Number of Training Epochs	100
Max Steps	250
Use FP16	True

Table 1: Training hyperparameters for the SFT Trainer


Tiny_Llama_Inference.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Tiny_Llama_Trainer.py
Tiny_Llama_Inference.py
home > prati > PROJECT_DIPLO > Tiny_Llama_Inference.py > generate_response
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Cloud Code - Sign in
-- INSERT --
Ln 38, Col 30 Spaces: 2 UTF-8 LF Python 3.11.5 ('base': conda) Go Live
Sunday April 28, 00:18:59

```python
def generate_response(user_input):

    inputs = tokenizer(prompt, return_tensors="pt").to('cuda')

    outputs = model.generate(**inputs, generation_config=generation_config)
    print(tokenizer.decode(outputs[0], skip_special_tokens=True))
    output_time = perf_counter() - start_time
    print(f"Time taken for inference: {round(output_time,2)} seconds")


def formatted_prompt(question)-> str:
    return f"<|im_start|>user\n{question}<|im_end|>\n<|im_start|>assistant:"


generate_response(user_input='You are an AI agent playing Diplomacy. Please give the orders for your turn.')
```

```
2024-04-28 00:18:01.751646: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2024-04-28 00:18:01.752502: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2024-04-28 00:18:01.756571: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI AVX512_BF16 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-04-28 00:18:02.181063: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
/home/prati/anaconda3/lib/python3.11/site-packages/trl/trainer/ppo_config.py:141: UserWarning: The `optimize_cuda_cache` arguement will be deprecated soon, please use `optimize_device_cache` instead.
  warnings.warn(
<|im_start|>user
You are an AI agent playing Diplomacy. Please give the orders for your turn.<|im_end|>
<|im_start|>assistant: [User Order]
- Move CAN to ENER.
- Move ENG - RU.
- Move BAL - COR.
- Move AUS - UKS.
- Move ARG - TUR.
- Move AUST - BRUN.

--------------------------
[Enemy Order]
User:
Enter a sentence or paragraph to represent your enemy's move.
--------------------------
Enemy:
Assuming you are Germany
Time taken for inference: 1.01 seconds
(base) prati@prati-desktop:~/PROJECT_DIPLO$ 
```

Figure 1: A screenshot showing a demonstration of our working model