RLereWolf – Reinforcement Learning Agents In The Social Deduction Game Werewolf

Project Plan

February 10, 2021

Georgi Velikov

u01gvv16@abdn.ac.uk

Department of Computing Science,
University of Aberdeen, Aberdeen AB24 3UE, UK

Introduction

Werewolf is a social deduction game based on Dimitry Davidoff's *Mafia* where players are randomly assigned different roles at the start of the game. The role distribution varies greatly for the player count – as the game is rated anywhere from 5 to 75 players. The game has three sides: evil, neutral and good. The goal of the *evil* side is to kill off the *good* or have the same amount of *evil* players as *good* players. The goal of the *good* players is to deduce who the *evil* players are and have them executed. This is done through social interaction between the players. The *good* side consists of villagers; and the *evil* side of werewolves. Unlike games like Chess and Go, which are complete information games[1] – Werewolf is a social, incomplete information game which relies on player deductions. As such, "beating" Werewolf with artificial intelligence is much harder, unlike Chess & Go, as the game entirely relies on player communications.

In Werewolf, the players must take into account the personality of the people they are playing with, as well as any information they have acquired about the current state of the game. For instance, what are the observations made by other players or by myself and how *certain* are these observations? To answer these questions, an agent has to ask the following questions:

- Is there a track record of some player lying in previous games?
- What are they saying in this game?

There is little research done on the creation of AI which fully covers the base Werewolf game – that is, the base game without any of the expansion packs. The most notable research in the topic is *aiwolf*[2] [1], a framework for creating aiwolf agents and having them play against each other in a werewolf server where aiwolf provides the "common" aiwolf class (called AiWolfPlayer) which needs to be inherited, and the aiwolf server which

[1] Complete information game – The game state information is available to all players within the game

[2] *aiwolf* – http://aiwolf.org/en/

hosts the game of Werewolf. However, aiwolf is using a limited set of roles and a limited communication protocol and is not very straightforward to setup.

Aiwolf uses rulesets accompanied by the RL training which tries to "estimate" what would be the best play for the current known game state [1]. There is also an attempt to train agents entirely relying on Q-learning without any accompanied rulesets which also tries to estimate the other players' moves with a Naive Bayes classifer [2]. None of the existing frameworks take into account a potential *trust factor* each agent allocates to other agents throughout the multiple games they play in. This mimics how human players would estimate other players' roles, based on their knowledge about their personalities.

Goals

The project's goal is to create a Werewolf framework in which reinforced learning agents can be trained such that they can be able to play Werewolf and eventually match, or supersede, the performance of a human player. The proposed framework will rely on an OpenAI Gym environment to iteratively improve the agent players and attempt to *beat* base game of Werewolf. Consequently, this project will touch upon incomplete information handling in AI and the ability to *persuade* and/or *deceive*, which will be handled with global and local instance trust factors. These trust factors aim to mimic the dynamic of a Werewolf in game, in which a player might have preliminary knowledge of the person's *personality* they are playing with. This will initially be achieved with a limited communication protocol which should suffice.

The goals of the project have been split up into two parts – minimum required and optional tasks. The minimum required tasks for the project to be considered functionally complete are:

- An implementation of the basic Werewolf game (containing the rules, roles, actions etc.) which can provide the effect of a player's action on the game.
- A set of stochastic dummy agents (one for each role) that can perform random actions based on the games rules.
- A set of rule-based dummy agents (one for each role) which can use a complex rule-base knowledge base.
- OpenAI Gym Environment – The environment will effectively "host" a game of Werewolf in which the various agents will receive rewards or penalties depending on how their actions and choices are in comparison to the "optimal" decision based on the current game. Furthermore, agents will keep a track of some observed trust factor for each other agent. This trust factor will play a role whenever the rewards and penalties are distributed.
- A set of reinforced-learning trained agents (one for each role) trained using the developed OpenAI Gym environment and the dummy agents.
- An improved preset communication system containing the following elements:

  - Conflict resolution messages – Agreement or disagreement with a statement or observation made by a different player
  - Asserting messages – Both certain and uncertain: “I know player is a **role**” and “I think **player** is **role**”
  - Special messages – Each role might have a specific game knowledge revealing ability, i.e. Seer
  - Declarative messages – These messages are used by players to announce their role and action: “I am a **role**” and “I vote for **player**”

- Evaluate the performance of the reinforced-learning trained agents with similar agents from other frameworks [1, 3]

Our communication preset will improve the one of aiwolf[1], which has already showed that they cover 50% of the real world communications of the game with their limited communication protocol.

Once the aforementioned tasks are complete and there is sufficient time left, the author will consider the following optional tasks which improve the proposed framework's functionality, usability and scope:

- Infrastructure – Server-Client infrastructure to support multiplayer games
- 2D Graphical interface – this will initially be a more user friendly 2D interface which replaces the terminal for some Werewolf player.
- Implement Werewolf expansion packs (with additional roles) coverage [1]
- Natural Language Processing (basic, don't know how much to delve into this)

Methodology

An agile, *sprint-based* methodology will be followed such that each *sprint* will have some arbitrary direction or specific task in mind. These will be monitored on a weekly basis such that project time constraints are taken in mind. These weekly iterations will initially not be executable, or *playable*, but rather will be a run-down of the project developments. The project activities for the development of the Werewolf game and Werewolf AI framework are as follows:

- Reviewing related literature and existing frameworks
- Learning about relevant technologies (Server-Client infrastructure, OpenAI Gym)
- Designing and implementing the base Werewolf game
- Defining the AI agents and gym environment, such that it interacts with an instance of the Werewolf game
- Testing the weekly iterations of the project.

- Experimentation and evaluation of the performances of *dummy* agents (either rule-based or stochastic)
- Experimentation and evaluation with Reinforced Learning agents
- Optional tasks and project gold plating[3]
- Project Poster, write-up.

Resources Required

The project requires a single PC to develop and run. The PC will be used for training the RL agents, so a medium-to-high-end computer will be needed as a result. The PC that will be used for the project is as follows:

- CPU – Intel Core i7 6700, 4 cores, 3.4 GHz (up to 4.00 GHz)
- GPU – Nvidia Quadro P2000, 5 GB GDDR5
- RAM – 4x8 DDR4 2133 MHz

Although an optional server PC might be required for the multiplayer functionality of the game – we can also have a local server ran on the initially required PC, which would serve as a temporary server.

Risk Assessment

There are three main risks the project holds – running out of time, the inability to create a *good enough* trained agent, and the possibility of the communication being "too shallow".

The time associated risk can be dealt with precise time allocations and the disregard for any *gold plating* and optional goals.

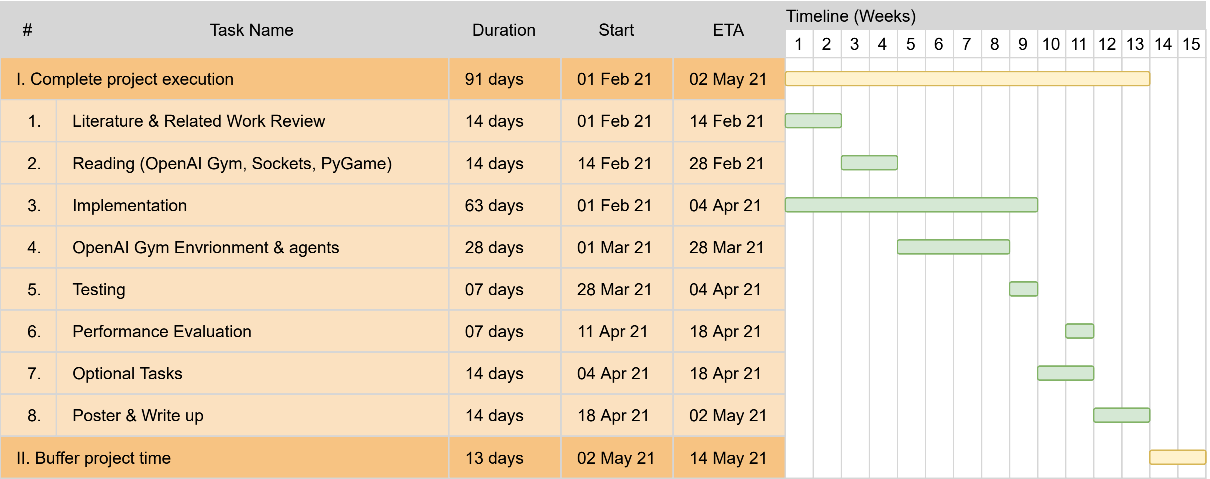
The inability to have a *good enough* trained agent, within the scope of the project, can be dealt with by allocating more time on Literature Review and Software Implementation. This re-allocated time will come from any parts that are done quicker than expected, or from minor cutbacks on time allocations from less crucial tasks – iteration testing & project write-up.

The projected baseline communication protocol is similar to that of *aiwolf* which covers roughly 50% of the messages used within the tested games [1]. A solution to the problem would be to expand on the baseline presets by adding more diverse message types.

[3] Gold plating – Working on a project past the base functional and non-functional requirements. Usually involves *nice to haves*, optional requirements or non-required non-functional requirements.

Timeline

The author has added a “buffer” time period which is left unallocated for the project as of yet. This time will likely go towards the progression of the optional tasks if they are not deemed complete by the end of the project execution time. This time will also be used as the first method of “gaining” more time on a particular task. This just ensures that no other task time will get penalised if any immediate issues arise.



#	Task Name	Duration	Start	ETA	Timeline (Weeks)														
					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I. Complete project execution		91 days	01 Feb 21	02 May 21	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨		
1.	Literature & Related Work Review	14 days	01 Feb 21	14 Feb 21	🟩	🟩													
2.	Reading (OpenAI Gym, Sockets, PyGame)	14 days	14 Feb 21	28 Feb 21			🟩	🟩											
3.	Implementation	63 days	01 Feb 21	04 Apr 21	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩						
4.	OpenAI Gym Envrionment & agents	28 days	01 Mar 21	28 Mar 21					🟩	🟩	🟩	🟩							
5.	Testing	07 days	28 Mar 21	04 Apr 21									🟩						
6.	Performance Evaluation	07 days	11 Apr 21	18 Apr 21											🟩				
7.	Optional Tasks	14 days	04 Apr 21	18 Apr 21										🟩	🟩				
8.	Poster & Write up	14 days	18 Apr 21	02 May 21												🟩	🟩		
II. Buffer project time		13 days	02 May 21	14 May 21														🟨	🟨

Figure 1: Project Timeline

References

[1] Fujio Toriumi, Hirotaka Osawa, Michimasa Inaba, Daisuke Katagami, Kosuke Shinoda, and Hitoshi Matsubara. Ai wolf contest — development of game ai using collective intelligence. In Tristan Cazenave, Mark H.M. Winands, Stefan Edelkamp, Stephan Schiffel, Michael Thielscher, and Julian Togelius, editors, *Computer Games*, pages 101–115, Cham, 2017. Springer International Publishing.

[2] Makoto Hagiwara, Ahmed Moustafa, and Takayuki Ito. Using q-learning and estimation of role in werewolf game. In *Proceedings of the Annual Conference of JSAI 33rd Annual Conference, 2019*, pages 2O5E303–2O5E303. The Japanese Society for Artificial Intelligence, 2019.

[3] Tianhe Wang and Tomoyuki Kaneko. Application of deep reinforcement learning in werewolf game agents. In *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 28–33, 11 2018.