Code Description of Team Sashimi

for the 3rd International AIWolf Competition

August 14, 2021

1 Development Environment

We used Java 11 to develop our agent. The classpath is `jp.ac.tsukuba.s.s2020602.SashimiRoleAssignPlayer`. The ZIP file of the source code contains the JAR file: `SashimiAgent.jar`. Our agent was built for Linux and we have not confirmed whether it works on the other OS's.

2 Overview of our agent

Our agent mainly consists of two modules: a module that estimates the roles of the other players and a module that decides the behavior (i.e., content of utterance and voting) each turn. For the role estimation, our agent uses fastText, a library for learning text classification, and take the approach of reducing the role estimation to the problem of text classification. This is an extension of the idea used by the cndl team, who won the Protocol division at the 4th AIWolf Competition in 2018.

For the decision of behaviors, our agent is based on the idea of game theory, especially Bayesian games, a class of games in which information about the types of players is not revealed. More specifically, first, the likelihood of roles of the other players is represented as a probability distribution, which is called the player's belief. Our agent updates its belief based on the utterances and votes of the other players. Our agents also gain a predetermined utility depending on who is expelled at the end of each day. Based on the belief and the utility function, our agent decides the behavior so as to maximize the expected utility.

3 How to Estimate Roles and Vote

Role estimation is considered as a text classification problem to label the text of the game log (including all the players' utterances and votes) with the answer to the question of who has each role. The cndl team's agent used a model that makes independent sequences of the utterances and votes for each agent, while our agent puts together all the players' utterances and votes into a single sequence in order to take account of causal relationships between utterances. We built a model for predicting who has each role from the point of view of an agent having each role in 5-player games and 15-player games. When evaluating expected utilities, we assume that all the agents (including not only ours but also the others') use a relevant model among them and decide whom to vote for according to the estimation. For example,

a player in the villager camp votes for the agent who is considered the most werewolf.

4 How to Decide What to Say

Our agent decides what to say as follows. First, it computes its belief by role estimation as a probability distribution on the combination of all roles in the game. In 15-player games, it limits its consideration to the key roles. For example, a villager player will think that there are 3 werewolves and 12 villagers in the game. Next, it enumerates an available choice of utterances. For example, if its role is seer, it can come out a seer (and report a divination result) or tell the suspicious agent to everyone. According to the belief, it computes the expected utility $EU_i(s)$ of each choice s by the following equation:

$$EU_i(s) = \sum_{\theta \in \Theta} \mu_i(\theta) u_i(\langle l, a \rangle, \theta)$$

where

- Θ is the set of all the combination of roles;
- $\mu_i \colon \Theta \to [0, 1]$ is the agent i's belief;
- l is the list consisted of each event in the game log; and
- $u_i(l, \theta)$ represents the agent i's utility when all the agents decide whom to vote for based on the game log l (according to the method described in Section 3) under the role combination θ.

After computing the expected utility for each choice, it chooses one that will give the maximum utility.