ICIC Express Letters
Volume **12**, Number **4**, April **2018**

ICIC International ©2018 ISSN 1881-803X
pp. **313–319**

A METHOD FOR DETECTING WEREWOLVES IN THE WEREWOLF GAME BASED ON VECTOR REPRESENTATIONS OF WORDS

Atsushi Ueno, Masaki Sakamoto and Tomohito Takubo

Graduate School of Engineering
Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan
{ ueno; takubo }@info.eng.osaka-cu.ac.jp; sakamoto@kdel.info.eng.osaka-cu.ac.jp

Received October 2017; accepted January 2018

Abstract. *The Werewolf game is a multi-player game based on conversation. It is one of new targets of game artificial intelligence. In the game, each player plays a role and aims for victory through communicating, deceiving, and detecting lies in messages from other players. In this paper, we propose a method for detecting werewolves based on players' conversation in natural language in the Werewolf game. In this method, each player at a certain point of a game is represented as a vector, called player vector, based on all his/her messages up to that point. It computes how likely each player is a werewolf using a classifier in the space of player vectors. We verify its effectiveness through experiments on logs of an online site of the game.*
Keywords: Werewolf game, Game artificial intelligence, Skip-gram model, Vector representations of words, Support vector machine, Neural network

1. **Introduction.** In recent years, *artificial intelligence* (AI) players outperform the best human players in major perfect-information extensive-form games such as chess, shogi, and the game of Go. Many researchers are now aiming to develop good AI players in more difficult games. The *Werewolf game* is one of such games. It is a multi-player game based on conversation, in which each player plays a role and aims for victory through communicating, deceiving, and detecting lies in messages from other players. It is not an extensive-form game: It has no predetermined shape of the game tree because all players can send messages at any time. Each player has infinite choices of what to say in a natural language. There are some secret conversations in which only limited players can participate. Because of these properties, it is very difficult for each player to predict the flow of a game. For this reason, search algorithms that are extremely useful in extensive-form games are useless in this game. A research project called the "AIWolf Project[1]" has been started and the competition is held every year since 2015. It is expected that development of good AI players in this game leads to understanding human intelligence for making groups, communities, or societies and getting along well in them through communication.

In the Werewolf game, players are divided into two camps: the werewolf camp and the villager camp. Players in the villager camp cannot identify who the werewolves are. A key factor for victory of the werewolf camp is that each werewolf pretends well to be a villager. A key factor for victory of the villager camp is to detect werewolves based on messages from other players.

In this paper, we propose a method for detecting werewolves based on players' conversation in natural language in the Werewolf game. This method can be used as a support tool for human players who need suggestions about identities of the other players. In this method, all words in all the messages from each player in a game are represented as

[1]http://aiwolf.org/en/

continuous vectors, which are learned by a skip-gram model [1]. The player at a certain point of the game is represented as a normalized vector based on the vectors of all the words in all his/her messages up to that point. We name it a *player vector*. In the space of player vectors, a classifier is learned to distinguish werewolves from villagers. We can compute how likely each player is a werewolf using this classifier. We try a *support vector machine* (SVM) classifier and a *neural network* (NN) classifier for being incorporated in our method. The original SVM algorithm was proposed as a linear classifier. SVM classifiers have been widely used as excellent nonlinear classifiers since Boser *et al.* [2] proposed a way of applying the kernel trick to the original one in 1992. NNs have been studied for a long time. They have been studied extensively around since a Google research team succeeded in training an NN to recognize cats in images in 2012 [3]. This paper is organized as follows. Section 2 describes the outline of the Werewolf game, the AIWolf Project, and other previous works related to the game. Section 3 describes the details of our proposed method. In Section 4, we verify its effectiveness through experiments on logs of an online site of the game. Section 5 concludes the paper and discusses the future work.

2. **The Werewolf Game.**

2.1. **Outline of the game.** In the Werewolf game, players are divided into two camps: the werewolf camp and the villager camp. Though players in the role of werewolves know who the werewolves are, players in the other roles cannot. The game proceeds in alternating day and night phases. In a day phase, all the players have a conversation with lies and decide one player to be executed by voting. All the werewolves also have a secret conversation to decide one player who they will kill at that night. At the end of the day phase, the execution is held. In the night phase, the werewolves kill a player. Then, those who survive proceed to the next day phase.

The villager camp wins if the villagers kill off all the werewolves. The werewolf camp wins if the werewolves kill enough villagers so that the numbers are even. Some villagers have special abilities: Each day, the seer can conduct divination on a player, the psychic can expose the identity of the dead, and the knight can protect a player from a werewolf attack. Some human player, the madman, belongs to the werewolf camp and aims for the victory of the werewolf camp.

2.2. **The AIWolf Project.** The AIWolf Project aims for encouraging development of good AI players in the Werewolf game. It has been developing the AIWolf server, which provides a game table for players and serves as the moderator of the game. It has also been developing a communication protocol between the players. In this protocol, the available vocabulary and sentence patterns are limited. Therefore, you cannot talk about everything you want. In addition, there are few options of wording. We think that it may be a critical feature in complicated communications, such as deceiving others or detecting lies. Therefore, we aim for detecting werewolves based on natural language conversation between human players rather than communications in this protocol.

Participants of the AIWolf Project create AI players which can play the game actually with other AI players. Our method cannot play the game by itself. At this point, we think that it can be useful as a support tool for human players.

2.3. **AI methods for the Werewolf game.** In the Werewolf game, it is extremely important for villagers to deduce who the werewolves are. Kajiwara *et al.* [4] proposed such a method for AI players in the AIWolf competition. In this method, each player is represented by several features that are important to judge if he/she is a werewolf. In the space of these features, this method classifies between werewolves and villagers using an SVM classifier. It is different from our method on the point that it detects werewolves based on conversation between AI players that uses the limited protocol. Moreover, the

important features are designed by human designers. We aim for autonomous extraction of important features from natural language conversation.

Lin *et al.* [5] proposed a set of inference rules for limiting candidates of the roles based on players' conversation in natural language. They have not described how to implement the *natural language processing* (NLP). This set of rules can decide whether each player has the possibility of being a werewolf or not. It cannot decide how likely he/she is a werewolf. On the other hand, the objective of our method is to decide it.

As an AI method with NLP, Hirata *et al.* [6] proposed a method for automatic tagging of messages in the Werewolf game among human players. The method can annotate messages with three types of tags: *revealing* oneself as some role, telling something about *execution*, and telling something about *divination*. These tags have only superficial information. The annotated messages may include lies. We aim for developing a method that can have a deep insight into messages for detecting lies.

3. Proposed Method.

We have developed a method for detecting werewolves based on players' conversation in natural language in the Werewolf game. In the method, firstly, all messages are divided into words. This process is necessary when handling a non-segmented language, such as Japanese. We use the morphological analyzer *MeCab*[2] with the neologism dictionary *mecab-ipadic-neologd* and a handmade slang dictionary for dividing messages in Japanese. Secondly, all words in all the messages from each player are represented as continuous vectors, which are learned by a skip-gram model. Thirdly, player vectors are computed at all times after sending a message based on the word vectors, each of which represents the sender of the message. Fourthly, all player vectors are classified between werewolves and villagers by a classifier. Finally, the method computes how likely each player is a werewolf based on the outputs of the classifier. We call the degree of likelihood of being a werewolf a *wolfiness score*.

3.1. Vector representations of messages.

We use the skip-gram model [7] for representing all words in all messages as continuous vectors. It is a model for learning word vectors using an NN. It places word vectors close to each other if the words are often used in similar contexts. By using it, word vectors have a simple mathematical property: Addition of word vectors can often produce meaningful results. For example, vec("Germany") + vec("capital") is close to vec("Berlin"). This model is suitable for using in our method, in which vector representations of messages and players are calculated in the space of word vectors as mentioned below. We use a new variant of the skip-gram model proposed by Bojanowski *et al.* [1], in which each word is represented as a bag of character n-grams. This model can take account of subword information, so it can convert rare words not in the vocabulary of the training data into vectors. The reason why we have adopted this model is that conversations in the Werewolf game include a lot of slang and new words. We use the zero vectors for representing any words that cannot be converted in this model.

The vector representation of each message is defined as the normalized sum vector of the vectors of all the words in the message:

$$\boldsymbol{m}_{p,i} = \frac{\sum_{j=1}^{n_{p,i}} \boldsymbol{w}_{p,i,j}}{\left\| \sum_{j=1}^{n_{p,i}} \boldsymbol{w}_{p,i,j} \right\|},$$

where $\boldsymbol{m}_{p,i}$ is the i-th message from player p in the game, $n_{p,i}$ is the number of the words in the message, and $\boldsymbol{w}_{p,i,j}$ is the j-th word in the message.

[2]MeCab ver. 0.996. http://taku910.github.io/mecab/

3.2. **Player vectors.** At the time of sending each message, a player vector of the sender is defined as the normalized sum vector of the vectors of all the messages sent from him/her up to that point:

$$\boldsymbol{v}_{p,i} = \frac{\sum_{j=1}^{i} \alpha^{i-j} \boldsymbol{m}_{p,j}}{\left\| \sum_{j=1}^{i} \alpha^{i-j} \boldsymbol{m}_{p,j} \right\|}, \tag{1}$$

where $\boldsymbol{v}_{p,i}$ is the player vector of player p at the time of sending i-th message in the game. Each player has a lot of player vectors, each of which represents him/her at the time of sending each message. $\alpha \in (0, 1]$ is the forgetting factor.

3.3. **Computation of the wolfiness score.** All player vectors are classified between werewolves and villagers by a classifier, and the wolfiness scores of all the players are computed based on the outputs of the classifier. We try an SVM classifier and a three-layer NN (3LNN) classifier for being incorporated in our method. In the SVM classifier, we use the soft-margin technique and the Gaussian kernel. In the 3LNN classifier, we use the softplus activation function in all neurons, softmax cross entropy as the loss function, and Adam [8] as the optimizer. The input layer has the same number of neurons as the number of dimensions of the player vectors and receives one of them. The output layer has two neurons, each of which indicates how likely the inputted player vector belongs to the werewolf camp or the villager camp.

Since voting for execution is held every day in the game, each player has to select another as a suspected werewolf every day. In judging each player on the i-th day, our method uses all his/her player vectors on the first to i-th day as training or test data. When using the SVM classifier, all the test data of each player are classified by a trained SVM, and the wolfiness score is defined as the percentage of being judged as a werewolf. When using the 3LNN classifier, output values are computed for all the test data of each player by a trained 3LNN, and the wolfiness score is defined as the mean of all the output values of the neuron for the werewolf camp. In the case of using the proposed method as a support tool, it can recommend a player with the greatest wolfiness score as an execution target.

4. **Experiments and Discussion.** In this section, we verify the effectiveness of the proposed method through experiments on logs of an online site of the Werewolf game, "WolfBBS: G[3]". We use the logs of fifteen-person games in the experiments, each of which includes three werewolves, one madman, and eleven other villagers. We have collected the logs of 900 fifteen-person games from Game #121 to Game #1688. The logs of 600 games have been selected randomly to compose the training data set, which is used for both learning word vectors by the skip-gram model and training the SVM and 3LNN classifiers. Since the madman belongs to the werewolf camp, the classifiers are trained to judge his/her player vectors as werewolves. The logs of the other 300 games compose the test data set. The length of a game changes according to the progression of the game. Table 1 shows the numbers of ongoing games on the second to ninth day in the training and test data sets. No game lasted more than nine days.

Table 1. Numbers of ongoing games

	2nd day	3rd day	4th day	5th day	6th day	7th day	8th day	9th day
Training data set	600	600	599	581	519	380	178	9
Test data set	300	300	299	288	248	186	82	7

[3]http://www.wolfg.x0.com/

In our method, the forgetting factor α in Equation (1) is set to 0.98. We have verified in preliminary experiments that the performance with this forgetting factor is a little better than that without forgetting. In the skip-gram learning, the dimension size of word vectors is set to 300, which is the same as that used in [1]. The other parameters for the skip-gram learning are set as follows: the size of the context window = 10; the number of negatives sampled = 10; the number of training epochs = 100; and the minimal number of word occurrences = 5. In the SVM classifier, the penalty parameters of the error terms for class *Werewolf* and *Villager* are set to 27.5 and 10.0, respectively. The reason of this difference is that the villager camp has 2.75 times as many members as the werewolf camp. The other parameters for the SVM classifier are set as follows: the variance of the Gaussian kernel = 50.0; and the limit on iterations within solver = 10000. The parameters for the 3LNN classifier are set as follows: the number of neurons in the hidden layer = 500; the step size in Adam $\alpha = 0.001$; the exponential decay rates for the moment estimates in Adam $\beta_1 = 0.9$ and $\beta_2 = 0.999$; and the small value parameter for the numerical stability in Adam $\epsilon = 10^{-8}$. Many of these parameters are set to commonly used values.

Table 2. Percentages of werewolves among players selected by the 3LNN classifier

	2nd day	3rd day	4th day	5th day	6th day	7th day	8th day
Setting #1	**46.62**	**54.85333**	**54.31438**	**50.71528**	**46.54839**	**42.63441**	**37.03704**
Setting #2	**46.76**	**54.80667**	**53.92642**	**50.72222**	**46.77419**	**43.07527**	**37.77778**
Setting #3	**47.3**	**55**	**54.05351**	**49.875**	**46.89516**	**43.32258**	**37.50617**
Setting #4	**46.63333**	**51.52667**	**54.92977**	**50.10417**	**46.58065**	**43.15054**	**38.29630**
Setting #5	**46.59333**	**55.31333**	**54.37458**	**50.20833**	**46.75**	**42.77419**	**36.88889**
Setting #6	**46.58667**	**54.84**	**54.42809**	**51.125**	**46.57258**	**42.48387**	**37.08642**
Setting #7	**46.98667**	**54.70667**	**53.72575**	**49.91667**	**46.40323**	**42.84946**	**37.23457**
Setting #8	*25.32*	**54.08**	**54.44816**	**49.875**	**46.64516**	**42.15054**	**36.69136**
Setting #9	*25.07333*	*28.28667*	*27.09030*	*26.63194*	*27.79839*	**42.79570**	*28.51852*
Setting #10	*25.81333*	*28.04667*	*27.27759*	*25.95139*	*27.06452*	*28.30108*	*28.29630*
#Success	**7**	**8**	**8**	**8**	**8**	**9**	**8**
Range	151-200	201-250	451-500	251-300	151-200	101-150	301-350
Average	**46.78286**	**54.39083**	**54.27508**	**50.31771**	**46.64617**	**42.80406**	**37.31481**
SD	**0.248650**	**1.129739**	**0.347463**	**0.444943**	**0.144681**	**0.337290**	**0.489915**

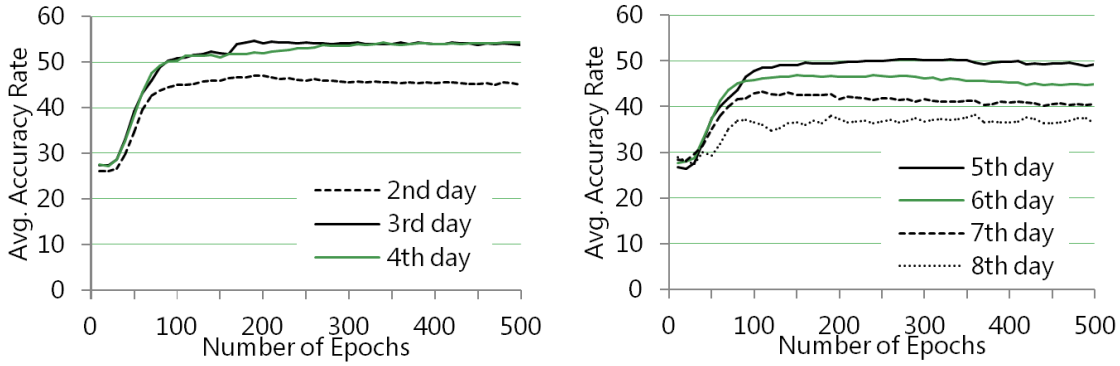
In the experiments, our proposed methods select a player with the highest wolfiness score on the second to eighth day. As shown in Table 1, we have so few data on the ninth day that we do not use them in the experiments. We calculate the percentages of werewolves among the selected players on the second to eighth day. For comparison, we calculate the percentages of the votes for werewolves among all the votes of villagers (not including the madmen) on the second to eighth day in the test data set. Since villagers always try to vote for a werewolf, these values are considered as the accuracy rates in the voting by the villagers. We also calculate the percentages of werewolves among survivors, which are equal to the accuracy rates for werewolf detecting in random voting.

In the experiments with the 3LNN classifier, we perform 10 runs with different random seeds, each of which is used to initialize a weight setting. Table 2 shows the percentages of werewolves among players selected by the 3LNN classifier. The values are the average percentages over 50 epochs after learning has converged. The 13th row shows the ranges of epochs for calculating the averages. The bold and shaded items indicate learning successes. The 12th row shows the numbers of learning successes among the 10 runs. The 14th and 15th rows show the averages and standard deviations of the average percentages over the success runs. The results can be divided clearly into two groups: success and failure. In the success group, the standard deviations are very small. All the data in the failure group are far below five times the standard deviations from the averages of

the success runs. It is easy to distinguish these two groups because the learning did not progress all the time in the failure runs. It is not difficult to find a good initial weight setting because the success rates of learning are high and the variances of performances in the success group are very small.

Figure 1 shows the learning curves averaged over the success runs. The accuracy rates are additionally averaged over ten consecutive epochs. The learning curves converge after approximately 150 epochs. Small performance degradations are observed in the results of the second, fifth, sixth, and seventh days. Small performance improvement continues until 500 epochs in the result of the fourth day.

Figure 2 shows the average accuracy rates for werewolf detecting on the second to eighth days. The values on the "3LNN" line are the same as those in the "Average" row in Table 2. Line "human" corresponds to the accuracy rates in the voting by the villagers in the test data set. They are the accuracy rates of real human players. Line "random" corresponds to the accuracy rates in random voting. The average result of our method with the 3LNN classifier is overwhelmingly superior to "human" on the second day. It is superior to "human" on the third day, close to "human" on the fifth and sixth days, and inferior to "human" on the other days. The average result of our method with the SVM classifier is superior to "human" on the eighth day, close to "human" on the seventh day, and inferior to "human" on the other days. Let us now consider a combined method that uses the 3LNN classifier until the sixth day and the SVM classifier since the seventh day. Such a method can be superior or close to the average human player other than on the fourth day.



(a) 2nd-4th days (b) 5th-8th days

FIGURE 1. Learning curves averaged over success runs

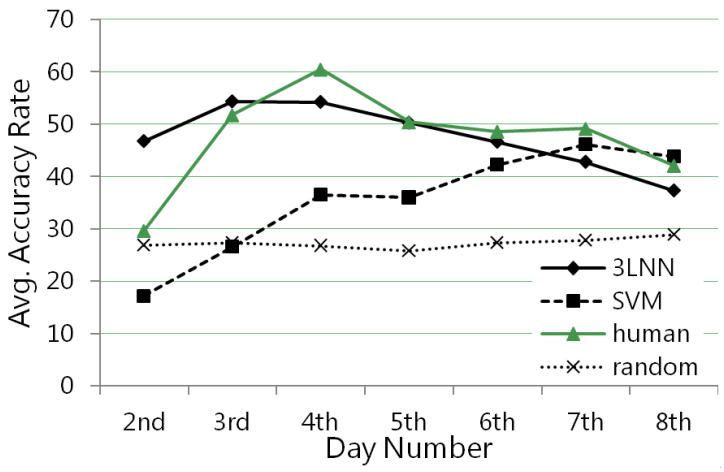


FIGURE 2. Average accuracy rates for werewolf detecting on the 2nd-8th days

At the end of this section, we estimate the effectiveness of our method as a support tool. For simplicity, let us consider the case of following the recommendations of our method about executions only on the second day. We choose the 3LNN classifier with the best average performance on the second day. Its average accuracy rate for werewolf detecting is 47.3% (Setting #3 in Table 2). In the test data set, the total winning percentage of the villager camp is 51.3%. In the cases where a member of the werewolf camp is designated for execution on the second day, it is 67.7%. In the other cases, it is 43.6%. Based on these percentages, we can estimate the winning percentage of the villager camp as follows: $0.677 \times 0.473 + 0.436 \times (1 - 0.473) \fallingdotseq 55.0\%$. By comparing it with the actual winning percentage (51.3%), if you just use this method on the second day, the winning percentage is expected to rise by 3.7%. The difference in the winning percentage between both the camps is expected to increase by about 4 times: from 2.6% to 10.0%. We think that this increase cannot be ignored.

5. **Conclusions.** In this paper, we have proposed a new method for detecting werewolves based on players' conversation in natural language in the Werewolf game. In this method, each player at a certain point of a game is represented as a vector, called player vector, based on all his/her messages up to that point. It computes how likely each player is a werewolf using an SVM or 3LNN classifier based on the player vectors. In the experiments of werewolf detecting, the average performance of our method with the 3LNN classifier is overwhelmingly superior to that of human players on the second day. A combined method can be superior or close to the average human player other than on the fourth day. As far as we know, no other AI methods can exceed the average human player in detecting werewolves in actual Werewolf games among human players. In addition, we have shown its effectiveness as a support tool for human players. We might go on to develop a method that can estimate a wolfiness score of a player based also on what is mentioned about him/her in the messages. We think that development of werewolf detecting methods is an important step towards a complete AI player of the Werewolf game as well as a powerful support tool for human players.

REFERENCES

[1] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching word vectors with subword information, *Trans. of the Assoc. for Computational Linguistics*, vol.5, pp.135-146, 2017.

[2] B. E. Boser, I. M. Guyon and V. N. Vapnik, A training algorithm for optimal margin classifiers, *Proc. of the 5th Annu. Workshop on Computational Learning Theory*, pp.144-152, 1992.

[3] Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean and A. Y. Ng, Building high-level features using large scale unsupervised learning, *Proc. of the 29th Int. Conf. on Machine Learning*, 2012.

[4] K. Kajiwara, F. Toriumi, M. Inaba, H. Osawa, D. Katagami, K. Shinoda, H. Matsubara and Y. Kano, Development of AI Wolf agent using SVM to detect werewolves, *Proc. of the 30th Annu. Conf. of the Japanese Society for Artificial Intelligence*, 2016 (in Japanese).

[5] Y. Lin, M. Baba and T. Utsuro, Generating a werewolf game log digest of inferring each player's role, *Proc. of the 15th IEEE/ACIS Int. Conf. on Computer and Information Science*, pp.807-812, 2016.

[6] Y. Hirata, M. Inaba and K. Takahashi, An automatic annotation method of speech act tag in Werewolf, *IEEE SMC Hiroshima Chapter Young Researchers' Workshop Proceedings*, pp.1-4, 2014 (in Japanese).

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, Distributed representations of words and phrases and their compositionality, *Advances in Neural Information Processing Systems*, vol.26, pp.3111-3119, 2013.

[8] D. P. Kingma and J. L. Ba, Adam: A method for stochastic optimization, *Proc. of the 3rd Int. Conf. on Learning Representations*, 2015.