Decoding the Chameleon Game

Tri Dang* (DePauw University; tridang_2025@depauw.edu)
Dat Nguyen* (DePauw University; datnguyen_2025@depauw.edu)
Hieu Tran (Purdue University; tran335@purdue.edu)
Brian Howard (DePauw University; bhoward@depauw.edu)
Sutthirut Charoenphon (DePauw University; sutthirutcharoenphon@depauw.edu)
DOI: 10.1145/3774399.3774403

Abstract

The Chameleon game is a challenging word association activity where players are given a secret word and must respond with words relevant to that secret word. It requires strategic thinking and deduction. The Chameleon must cleverly guess the secret keyword in this game while avoiding suspicion. Our research aims to create an advanced artificial intelligence (AI) model that can play the Chameleon game from both perspectives: as the Chameleon and as a Human. This AI is designed to guess secret keywords based on the information provided by the players, choose the best strategies to avoid detection as the Chameleon, identify and vote for the most suspicious players when acting as a Human, and learn from previous games to improve its performance.

Introduction

Word association games, where players are given a word (the cue) and must provide the first word that comes to mind (De Deyne, Navarro, Perfors, Brysbaert, & Storms, 2019) have gained significant traction due to their engaging and strategic nature, offering both entertainment and cognitive benefits. These games, which involve players generating words in response to given cues, enhance various cognitive functions such as attentional control, working memory, and problem-solving (Ornaghi, Brockmeier, & Gavazzi, 2011). Research shows that games (that is, chess and word association) can improve strategic thinking, memory, and creativity (Martinez, Gimenes, & Lambert, 2023)(Mercier & Lubart, 2021). While traditional word association games provide valuable cognitive and social benefits, they often rely on in-person interactions, which can be hindered by social isolation or geographical distance. Although networked online play can partially address these challenges, finding a sufficient number of players to participate simultaneously remains a significant obstacle.

To address this, we developed an AI model for the Chameleon game, a variation of word association where players must deduce a secret keyword while the Chameleon blends in by providing clues. By integrating techniques (e.g. Word2Vec, GloVe embedding, Naive Bayes, and Feed Forward Neural Network), the AI will learn from past gameplays and adapt to players' strategies over time. This approach not only replicates the interactive challenge of the original game but also creates a dynamic and evolving system that offers increasingly sophisticated gameplays, bridging social gaps and enhancing cognitive engagement.

Related Works

In board games, AI has demonstrated remarkable advancements through mastering complex games like Go, Chess, Poker, and Shogi. In "Mastering the game of Go without human knowledge", Silver et al. demonstrated how deep reinforcement learning can achieve superhuman performance through self-play (a method where an AI trains by competing against itself, learning and improving through repeated games), and highlighted AI's potential in strategic environments (Silver et al., 2016). Similarly, Silver and Hubert demonstrate that general reinforcement learning algorithms using self-play and extensive training enable AI to challenge humans in Chess and Shogi (Silver et al., 2018), while Brown and Sandholm developed Pluribus, an AI model that achieved superhuman performance in six-

*These authors contributed equally.

player no-limit Texas hold 'em poker (Brown & Sandholm, 2019). These studies highlight AI's transformative impact on strategic board games, demonstrating how self-learning algorithms achieve unmatched proficiency through rigorous training.

In word association games, AI applications are increasingly proving their values. Shi et al. used a black-box model to predict the difficulty of word games like Wordle, showing how AI can enhance players' experience and game design by analyzing challenge complexity (L. Shi, Chen, Lin, Chen, & Dai, 2023). Additionally, research by Richards and Amir focuses on opponent modeling in Scrabble (Richards & Amir, 2007). Their work investigates how AI can strategically predict and respond to opponents' moves, providing deeper insights into game dynamics and improving competitive play. In another research, Hasan and Gan developed an unsupervised adaptive brain-computer interface that improves gameplay in Hangman by learning from user brain signals (Hasan & Gan, 2012). Their system demonstrated enhanced accuracy and efficiency compared to non-adaptive methods. Together, these studies exemplify the growing role of AI in understanding and mastering word association games, from predicting game difficulty to modeling opponents' strategies.

While AI models have been applied to various word-association board games, little to no prior work has focused on modeling **The Chameleon**. Designed by Rikki Tahta and published in 2017 by Big Potato Games, **The Chameleon** is a social deduction word-association game where one player (the Chameleon) must bluff to conceal their lack of knowledge about a target word, while others attempt to identify them. The game's intricate player interactions and strategic deception introduce significant challenges for AI simulation and analysis. In this work, we address these challenges by employing neural networks to model the complex interactions inherent in The Chameleon. Specifically, we focus on developing a model that can effectively act as the Chameleon, blending in and successfully deceiving the opponents. Thus, our work not only applied to this game specific, but has the potential to model highly-complex interaction between humans in a social setting.

About the Chameleon game

Game Rules

The Chameleon game is designed for 4-8 players. The Game Components are shown in Figure 1. Starting from the Setup phase, players are randomly assigned as either Humans or Chameleons. All players receive a Code Card, except one who gets the Chameleon Card. A Topic Card with sixteen words relating to a theme is placed face-up. Humans use a grid reference from the Code Card to get a secret word.

Next, in the Give attributes phase, players take turns giving one-word clues (attributes) related to the secret word. Humans players will give attributes relevant to the keyword, while the Chameleon, who doesn't know it, needs to guess attributes based on Human clues.

After all clues are given, the game comes to the Voting phase, where Humans vote to identify the Chameleon. If the Chameleon is identified (Voting True), the game comes to the Guessing Secret Keyword phase, where the Chameleon gets one chance to guess the secret word. A Correct guess (Guess True) means a win for the Chameleon, while an incorrect guess (Guess False) means a win for the Humans. If the Chameleon is not identified (Voting False), the Chameleon wins the round. The game is played over multiple rounds, with total points determining the overall winner.

In this study, we studied a six-player version under the food theme. The six-player setup balances the game's complexity, providing enough interaction to challenge the Chameleon while keeping the deduction process engaging for Humans. The food theme works well because it draws from everyday experiences, making it easy for players to offer specific, recognizable attributes like flavors, textures, or common dishes. Other themes, such as animals, movies, colors, geography, or sports, can also be engaging options; however, their accessibility may vary based on players' knowledge and interests, particularly for younger players. The example of a play is shown in Appendix A.

Versions of Chameleon Games

The Chameleon game, a social deduction board game, has multiple versions, each offering unique gameplay elements. In the simple version, players don't have time to think before or during the Giving Attributes phase. As a result, subsequent players don't have time to consider previous players' clues and need to prepare a word before this phase. Voting in this version is based on identifying which attributes seem less relevant to the keyword.

In the complex version, during the Giving Attributes phase, players have time to think carefully before declaring unique attributes. Consequently, they can change their attributes or strategy based on previous players' clues. Voting considers various aspects, such as player behavior and the relevance of attributes across multiple keywords. In this research, we chose to focus on the harder version of the game, as mastering the more challenging aspects implies proficiency in simpler versions.

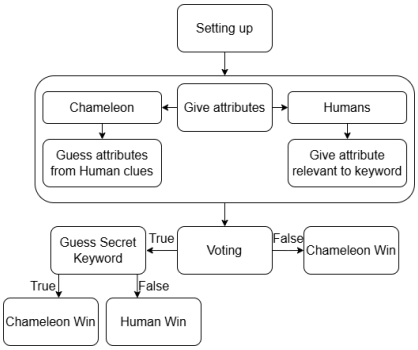

Figure 1: Game Component Diagram

Giving attribute as Chameleon

The Giving attributes phase is a key moment for the Chameleon, demanding both linguistic finesse and strategic forethought. Following the setup phase, where player roles and the secret word are determined, each participant must offer an attribute that upholds their credibility and furthers their strategic aims.

Attribute Representation in the Database

The database for the Chameleon game is structured to store attributes. Their relevance rate to each keyword is built on the idea of the Word2Vec model, which represents words in vector form (Jatnika, Bijaksana, & Suryani, 2019). A vector represents attributes in a 16-dimensional space, where each element indicates the relevance of that attribute to the corresponding keyword. When adding an attribute to the database, the default vector will be initialized with zeros, indicating no relevance to any keyword recorded. After matches, as the model learns from gameplay data, these vectors can be updated to reflect the true relevance of each attribute to the corresponding keywords.

For example, "Attribute 1" is initially added with a zero vector, indicating no relevance to any of the keywords. However, if "Attribute 1" is found to be relevant to "Pizza" and "Cake," the relevance value for "Pizza" and "Cake" in the vector will be adjusted accordingly. As the game progresses and more data is gathered, these relevance values will continuously refine, allowing the model better to understand the relationships between different attributes and keywords. This helps the technique make more informed decisions during gameplay, improving its ability to identify the keyword accurately or blend in as the Chameleon.

Getting and Handling Input Attributes

When it is the Chameleon's turn, the model will take the record of attributes provided by previous players as input to the model and, at the same time, use the database as a reference system to make predictions. When the AI model encounters a new attribute not present in the database, it will create a new entry for this attribute. This approach ensures that the model can handle unknown attributes and start with a neutral baseline. On the other hand, if the attributes already exist in the database, the AI model will retrieve the pre-existing vectors for these attributes.

Finding Attributes Intersection

After compiling the list of attributes given by the players, the AI model uses the data in the database to determine relevance. In this database, a value of 0 indicates no relevance between the meanings of attributes and keywords, while a value greater than 0 indicates at least some relevance between them. The model then searches for keywords relevant to all the attributes other players provide. If the AI

model cannot find any keywords that are relevant to all attributes (possibly due to missing data or because the combination of relevant keywords and attributes has never appeared in previous games), then it will progressively reduce the number of attributes it considers, checking keywords relevant to all but one of the previous players' attributes, then all but two attributes, and so on. This process continues until the AI model identifies at least one keyword that satisfies the requirement. By using this adaptive strategy, the AI ensures that it can always find at least a suitable keyword, even when the data is incomplete or when encountering novel combinations of attributes and keywords. This approach helps the AI maintain robust performance and effective gameplay.

To illustrate how the Chameleon identifies the intersection of attributes, consider the scenario where the Chameleon is in position 4, requiring it to analyze the previous three attributes. These attributes are represented as vectors, with each position corresponding to a specific keyword and the value indicating the relevance of the attribute to the keyword.

The model compares the vectors at each position to find the intersection as shown in Table 1. The intersection occurs where all three attributes share a common keyword, indicated by non-zero values across the vectors at that specific position. In this example, the keywords with index 2, 4, 6, 7, and 12 are identified as the intersection, where the Chameleon recognizes shared relevance among the previous attributes. This helps the Chameleon identify potential keywords that align with the input from previous players.

Keyword ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Attribute 1	2	1	0	3	0	2	3	0	1	2	0	3	0	1	0	3
Attribute 2	1	2	0	3	0	1	2	0	0	0	3	2	3	0	1	2
Attribute 3	0	3	2	1	1	3	2	3	0	1	2	3	2	3	1	0
Intersection		x		x		x	x					x				

Table 1: Attribute vectors and Intersection

Calculating Attributes Probability

After finding the attributes' intersection, we obtain a list of keywords that relate to all attributes from the previous players (the list of intersection keywords). Before calculating the probability, we set to zero any data that does not represent the relevance of an attribute to a keyword in the list of intersection keywords. This process is applied to all attributes in the database, ensuring that only the relevant values corresponding to the identified intersections are kept.

In the Table 1, Attribute 1, 2, and 3 is represented by the vectors. The intersection was identified at index 2, 4, 6, 7, and 12. In the new table, only the values at these positions are retained, while all other values are set to zero as shown in Table 2.

Keyword ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Attribute 1	0	1	0	3	0	2	3	0	0	0	0	3	0	0	0	0
Attribute 2	0	2	0	3	0	1	2	0	0	0	0	2	0	0	0	0
Attribute 3	0	3	0	1	0	3	2	0	0	0	0	3	0	0	0	0
Intersection		x		x		x	x					x				

Table 2: Modified attribute vectors

This step helps avoid data distractions that might come from attributes not in the intersection column but with high weight. Such data can significantly reduce the accuracy of the intersection column, which affects both the probability calculation and the subsequent spreading step. This process involves creating a new table where values irrelevant to the maximum number of available attributes are set to zero. By doing this, the model focuses solely on the relevant data, reducing potential distractions from unrelated attributes. This refined dataset enhances the technique's ability to make precise predictions and blend in effectively as the Chameleon.

We calculate the probability of relevant attributes for each keyword to account for the relevance of attributes to keywords and avoid interference from irrelevant data. This helps ensure that only significant data is considered in the spreading calculation. The probability for each keyword is determined by dividing the relevance value for that attribute by the sum of all the relevance values in the row (after masking out those not in the intersection).

After calculating the probability for each cell, we obtain a table showing the relevance of each attribute to each keyword as a percentage. This table helps to understand which keywords are more likely to be associated with the given attributes based on these probabilities. The percentage reflects the proportion of attributes relevant to each keyword compared to the total number of attributes available. Keywords with a relevance percentage

greater than 0.01 are included in the spreading calculation. This threshold ensures that only keywords with a meaningful number of relevant attributes are considered, thus reducing noise and improving the accuracy of keyword selection. For example, after applying the probability formula to Attribute 1 in Table 2, the resulting vector is shown in Table 3

Keyword ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Attribute 1 Probabilities	0	0.08	0	0.25	0	0.17	0.25	0	0	0	0	0.25	0	0	0	0
Sufficient relevance		x		x		x	x					x				

Table 3: Attribute 1 probabilities

Following the rule above that considers an attribute to be relevant to a keyword if its percentage exceeds 0.01, we observe that there are five values meet this criteria. Therefore, the spreading of attribute 1 corresponds to five keywords. After calculating the keyword spreading for all attributes, the table is then sorted by the number of keywords each attribute spreads, highlighting the significance of each attribute in the selection process.

To avoid predictable patterns for other players, the AI randomly selects between attributes with equal relevance within the range of intersection (spreading). This randomness ensures that the AI's choices are not easily anticipated by other players, maintaining the element of surprise and making it more challenging for Humans to guess who is Chameleon.

Updating the Database

After the match, the model records the attributes given by Humans and updates the database accordingly. For each attribute provided by the players that is relevant to the keyword used in the match, the corresponding value in the database increases by 1. This increment represents an increase in the relevance of that attribute to the keyword, indicating that real players are more likely to choose that attribute when considering the keyword. This continual learning process allows the model to refine its understanding of the relationships between attributes and keywords. By incrementally updating the database after each match, the technique becomes better at making accurate guesses and effectively blending in as the Chameleon. This adaptive learning enhances the model's performance and its ability to participate in the game in a more human-like manner.

Voting

After all players finish giving the attributes, the game comes to the Voting phase as shown in Figure 1. In this phase, each player is presented with key information that includes the six attributes provided by all six players, one of whom is the Chameleon. Five Humans will know what the secret keyword is. These attributes are revealed in a specific order, corresponding to the order of the players. Each Human must assess these attributes' relation to the keyword to determine who might be the Chameleon. The Chameleon, however, operates under different conditions. In the "Guessing Keyword as Chameleon" phase, the Chameleon can deduce the secret keyword based on the attributes provided by the other players. This deduction occurs before the Voting phase and does not rely on information from the Voting process itself. During this phase, the Chameleon uses the attributes given by other players to infer the keyword and strategically craft responses to blend in with the other players.

To enable Humans to identify the Chameleon and allow the Chameleon to use the network during voting to decrease suspicion and shift focus away from themselves, we employ a Neural Network (Guresen & Kayakutlu, 2011). The network layers are described in the following subsections, and the structure is shown in Figure 2.

Input Layer

The Neural Network's input layer is designed to process the information available to each player. Specifically, the player is given six attributes—one from each of the six players, including the Chameleon—and a secret keyword. The first step in this process is to calculate the distance between each of the six attributes and the keyword using cosine similarity (Chowdhury, 2010).

These distances are essential as they measure how closely related each attribute is to the keyword, reflecting the likelihood that the attribute was chosen by someone who knows the keyword or by the Chameleon attempting to blend in. The resulting six distances are fed into the Neural Network as input features.

Neural Network	Training Min. Loss	Training Max. Accuracy	Testing Accuracy
6-1-6+DO(0.2)+BN	1.70	29.58	13.30
6-2-6+DO(0.2)+BN	1.64	36.77	15.14
6-1024-6+DO(0.2)+BN	1.46	57.83	16.22
6-1024-512-6+DO(0.2)+BN	1.39	64.83	18.34
6-1024-512-256-6+DO(0.2)+BN	1.30	75.29	25.32
6-1024-512-256-128-6+DO(0.2)+BN	**1.24**	**80.84**	32.75
6-1024-1024-1024-1024-6+DO(0.2)+BN	1.45	58.61	53.30
6-1024-512-256-512-1024-6+DO(0.2)+BN	1.39	64.43	**64.25**

Table 4: Comparison between different Neural Network

Figure 2: Illustration of Neural Network

Hidden Layer

The Neural Network is designed with multiple layers, each playing a critical role in processing the input data. The network starts by receiving six inputs from the input layer, which represent the calculated distances between the attributes and the keyword. The hidden layers consist of 1024 neurons each. As the data passes through these layers, it is progressively refined, allowing the network to capture increasingly nuanced patterns and relationships within the data (Uzair & Jamil, 2020). The ReLU (Rectified Linear Unit), $\max(0, x)$, is used as the activation function between these layers (Arora, Basu, Mianjy, & Mukherjee, 2018; Nair & Hinton, 2010).

ReLU effectively introduces non-linearity into the model, which is essential for solving complex classification problems. It allows the network to pass only positive values to the next layer while setting negative values to zero, enabling the network to learn diverse patterns without the issue of vanishing gradients that can occur with other activation functions, such as the sigmoid function. Unlike the sigmoid function, which compresses outputs between 0 and 1 and is typically used in binary classification, ReLU allows the network to learn a broader range of patterns by not restricting the output values (Pratiwi et al., 2020). To further enhance the network's stability and performance, we incorporated Batch Normalization (BN) between the hidden layers (Bjorck, Gomes, Selman, & Weinberger, 2018). BN normalizes the input to each layer within a mini-batch, reducing internal covariate shift and speeding up the training process. This normalization enables the use of higher learning rates, leading to faster convergence and improved overall accuracy. Additionally, BN acts as a form of regularization, helping to mitigate overfitting by ensuring that the network's learning process is more robust.

Despite these improvements, overfitting remained a concern, particularly as indicated by the significant difference between training and test accuracy (as shown in Table 4). To address this issue, we employ Dropout with a rate of 0.2. The dropout randomly deactivates 20% of neurons during each training iteration, reducing the bias, and enhancing the generalizability of the model (Baldi & Sadowski, 2013). This technique effectively reduces overfitting, leading to improved performance on unseen data. In parallel with Dropout, we also applied L2 Regularization across the entire model. L2 regularization adds a penalty proportional to the square of the magnitude of the model parameters, encouraging the network to maintain smaller weights and avoid overfitting (G. Shi, Zhang, Li, & Wang, 2019). While L2 regularization provides a baseline level of regularization for all models, the introduction of Dropout further enhances the model's ability to avoid overfitting by promoting diversity in the learned representations.

After extensive testing and tuning, this specific network configuration, consisting of 6 in-

put neurons, 4 hidden layers with 1024 neurons each, and 6 output neurons, using ReLU as the activation function, L2 regularization, dropout (rate 0.2), and batch normalization, was found to provide the best performance for the task, as shown in Table4. This setup enables the network to effectively analyze the input data and make accurate classifications, whether for identifying the Chameleon or helping the Chameleon reduce suspicion during gameplay.

Output Layer

The output layer of the Neural Network is a multi-class classifier. It generates a prediction that identifies which player is most likely to be the Chameleon. The prediction is represented as one of six possible integers corresponding to the players in the game, with the model's objective being to accurately predict the player most likely to be the Chameleon.

In this model, we use Cross Entropy Loss as the loss function because it is the most effective and commonly used method for multi-class classification problems (Hinton, Osindero, & Teh, 2006). Cross Entropy Loss measures the performance of the classification model whose output is a probability value between 0 and 1. It calculates the difference between the actual label and the predicted probability distribution, penalizing predictions that are further from the true label. This loss function is particularly suitable for multi-class classification because it considers the probability distribution over all classes, ensuring that the model not only predicts the correct class but also assigns low probabilities to incorrect classes. By minimizing Cross Entropy Loss, the model is trained to provide more accurate and confident predictions across multiple classes, making it an essential component in developing a reliable AI model for tasks like identifying the Chameleon in the game.

Guessing keyword as Chameleon

Suppose the Humans identified exactly who the Chameleon is in the Voting phase (Voting True in Figure 1), the game comes to the Guessing keyword as Chameleon phase. In this phase, the Chameleon must guess the keyword based on all the attributes provided by the Humans. The process of finding the intersection of these 5 attributes is similar to how the attribute intersection is determined in the "Giving Attributes as the Chameleon" part. This process results in a table showing how many attributes intersect at each keyword.

After that, we calculate attribute probabilities, particularly in the context of the "Giving Attributes as Chameleon" step. The database is refined by setting any data that is below the highest number of intersecting attributes to zero. Subsequently, the refined dataset is utilized to compute the probability table for the 5 attributes. After determining the probability of each attribute given a relevant keyword, the Naive Bayes classifier—a simple learning algorithm that utilizes Bayes' rule along with the strong assumption that the attributes are conditionally independent given the class—is employed to estimate the probability of each keyword being the secret one (Webb, Keogh, & Miikkulainen, 2010). Initially, each keyword $P(k)$ is set to $\frac{1}{16}$. For each keyword k, the model calculates the conditional probability of each attribute A_i given the keyword: $P(A_i|k)$. Using the Naive Bayes formula, the combined probability of all attributes given the keyword is determined. Specifically, the model computes (Alpaydin, 2020):

$$P(k|A_1, \ldots, A_n) = P(k) \cdot \prod_{i=1}^{n} P(A_i|k) \quad (1)$$

The initial probability will be $P(k) = \frac{1}{16}$, since we have 16 keywords, and $A_n = A_5$ since we only have 5 attributes. To avoid underflow, which occurs when a calculation gets so close to 0 that the computer treats it as 0, we take the log of both sides of the formula (Haarhoff, Kok, & Wilke, 2013):

$$\ln(P(k|A_1, \ldots, A_5)) = \ln\left(\tfrac{1}{16}\right) + \sum_{i=1}^{5} \ln(P(A_i|k)) \quad (2)$$

After calculating the naive probabilities of the 16 keywords, the Chameleon will guess the keyword with the highest probability as the secret word for the game. If more than one keyword has the same Naive Bayes probability value, the Chameleon will randomly select one

among them. This approach ensures that the Chameleon makes an informed guess based on the calculated probabilities while also introducing an element of unpredictability.

Training Model

In the training phase, our best performance model will interact with five bots playing as Humans. Each bot is designed to give attributes to keywords using method described in the next section, and the voting strategy will have strategies similar to our best performance model in the Voting section above.

Training Database

We first used the Global Vectors for Word Representation (GloVe) model to generate an attribute list (GloVe list) (Pennington, Socher, & Manning, 2014). GloVe model captures semantic relationships between words by embedding them in a continuous vector space, where the distance between vectors reflects their semantic similarity. Words with similar meanings or contextual usage are positioned closely, enabling the identification and quantification of relationships between terms.

Using the GloVe model, we calculate the cosine similarity between each attribute in the GloVe list and each of the 16 keywords (Mikolov, Chen, Corrado, & Dean, 2013). Cosine similarity values close to 1 indicate high similarity, while values near 0 represent minimal relevance. Starting with attributes that have a cosine similarity of at least 0.50 with the keywords ensures that the attributes have at least moderate semantic similarity to the keywords, avoiding those that are too dissimilar. This threshold balances diversity and relevance, ensuring attributes remain distinct without drifting too far from the keywords' meaning. We then divided these attributes into six pools based on the cosine similarity value. Table 5 represents keywords with associated cosine similarity values.

In addition, we eliminate all attributes that are either too similar to a keyword or have a singular form already present. This ensures that the remaining attributes have a distinct semantic distance from the keywords and other attributes, allowing them to provide meaningful and diverse information. The higher the pool number, the more relevant the attribute is to the keyword, indicating a stronger semantic similarity. However, classification stops at Pool 6 because cosine similarity scores above 0.80 are rare for some attribute-keyword pairs, making it impractical to create additional pools beyond this threshold.

To determine which attributes the bots provide in the database, we apply a random sampling method that uses weights based on cosine similarity scores. Words with higher cosine similarity scores will be assigned greater weights, increasing their probability of being selected from the pool. This ensures that data points are appropriately chosen according to the similarity metrics.

The weights for each pool are determined using the inverse cosine function ($\cos^{-1}$) of the cosine similarity scores, given in degrees as can be seen in Table 5:

Pool	Cosine Similarity	$\cos^{-1}(x)$	Weight	Pool Percentage
1	0.50	60.0°	18.90	13.90%
2	0.55	56.3°	20.14	14.86%
3	0.60	53.1°	21.36	15.76%
4	0.65	49.5°	22.91	16.90%
5	0.70	45.6°	24.87	18.34%
6	0.75	41.4°	27.39	20.20%

Table 5: Weight of Pool

This weighting ensures that the most semantically relevant words (those in higher pools) are more likely to be chosen, reflecting their closer relationship to the keywords. By employing this method, the analysis effectively leverages the GloVe model's ability to capture nuanced semantic relationships, enabling a more precise and contextually aware interpretation of textual data.

To determine which attribute to use, we employ a two-layer random selection process.

Layer 1: Distribute Words into Pools
Attributes are first distributed into different pools. Within each pool, attributes are chosen randomly, with each having an equal probability of appearing.

Layer 2: Randomly Select Pool
A pool is then randomly selected based on their weights. The numerator is the weight of a pool, and the denominator is the total weight of all pools. The corresponding percentage of being selected for each pool is shown in Table 5.

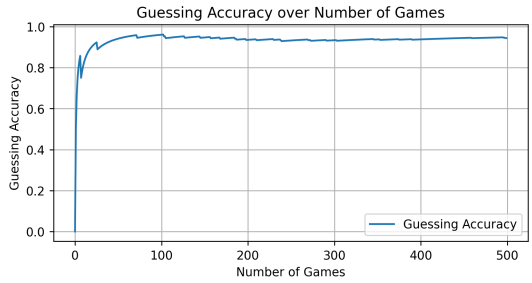

Figure 3: Chameleon Guessing Accuracy over Time

If two attributes are identified as similar, the process is repeated to generate a new list of six random keywords. This revised list is then distributed to five bots, ensuring that the final selection benefits from a combination of algorithmic randomness and human judgment.

By using the algorithm described above, we enhance the performance and reliability of the model by conducting a random sampling process 30,000 times to generate a diverse set of samples. This extensive sampling is designed to capture various data dimensions, ensuring that the model is exposed to a wide range of scenarios and variations. Figure 3 shows how the accuracy of guessing the secret word as the Chameleon changes over time.

The Chameleon model achieves a high guessing accuracy rate of over 92% with the training dataset, demonstrating its effectiveness in predicting keywords. Initially, the model's win rate starts low but rapidly climbs to over 90% as it gains more experience through the training process. Even after 1,000 games, the win rate keeps fluctuating slightly around this high value, indicating consistent performance. The large-scale sampling process further validates this performance, ensuring consistent and reliable accuracy across diverse data conditions.

Training Neural Network

Although we generated data from 30,000 games in the training database process, we used only the last 10,000 as training data for the neural network. The first 20,000 games were primarily used for populating the database, providing a robust foundation for the model's training. The goal was to ensure that the database captured a wide range of scenarios, player interactions, and possible outcomes, thereby enhancing the model's ability to generalize effectively.

Following this initialization, the model was trained across ten sessions, with each session consisting of 400 epochs. Figure 4 corresponds to the tenth training session, reflecting the model's performance after it had undergone significant refinement through the preceding nine sessions. This approach allowed the model to be improved progressively, minimizing loss and maximizing accuracy with each subsequent session. The rigorous training process, supported by a well-initialized and diverse dataset, contributed to the model's robust performance in predicting the secret word as the Chameleon, achieving high accuracy.

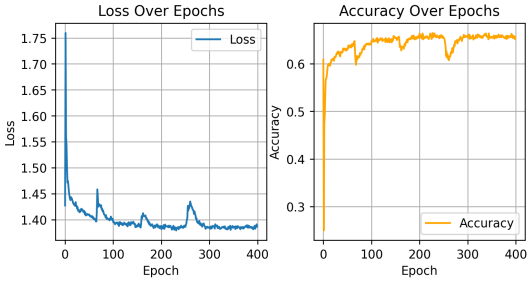

Figure 4: Loss graph and Accuracy graph

Testing Model

After the training phase, we created an additional 2,000 games for testing. In these test games, the Chameleon achieved a win rate of over 90% and was voted out in 64% of the games. The bot selects a keyword based on the rule outlined in the training section. The Chameleon's high win rate demonstrates its effectiveness in a controlled environment where the bots, due to their predictable and informative cues, allow the Chameleon to blend in and avoid detection more easily. However, this success is expected to drop significantly in a different configuration—specifically, when five people are playing with our model. People, with their less predictable and more varied hints, introduce greater complexity and additional variables, making it more challenging for the Chameleon to remain undetected.

The moderate voting rate of 64% likely results from the stochastic nature of attribute selection among the bots, introducing a degree of randomness that may assist the Chameleon in evading detection. The interplay of these

factors is depicted in the Figure 5, illustrating how the Chameleon's win rate and the effectiveness of the voting process evolve under different conditions.

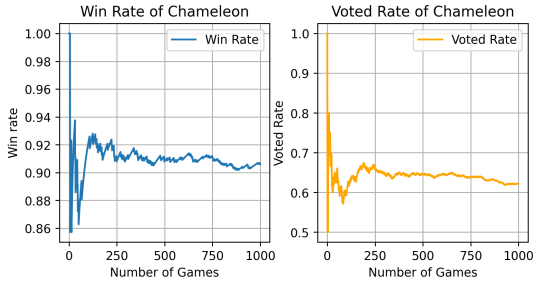

Figure 5: Chameleon Win and Voted Rates after Training

Conclusion

Developing the AI model for Chameleon Games has revealed several strengths and areas for improvement. A significant challenge was the lack of readily available, real-world datasets recorded after each game. This limitation highlights the need for future efforts to expand the dataset by collecting more real game data, enhancing the model's ability to simulate and predict player behavior more accurately.

Despite these challenges, the model demonstrates promising capabilities. The model is designed to learn and adapt after each match, allowing it to refine its accuracy and effectiveness over time continuously. With these strengths and a commitment to addressing the identified limitations, the AI model has the potential to become a robust and versatile tool for simulating and analyzing gameplay in Chameleon Games and beyond.

Additionally, the current implementation uses a simple neural network architecture. Future work should focus on exploring more sophisticated neural network models to improve the model's performance and adaptability.

Code

Source code available at https://github.com/DatNguyen2003/EAAI_2025.

Appendix A: Chameleon Game Simulation

Game Theme: Food
Secret Word: Salad

System: Gathering clues

- **Player 1 (Human)**: "coleslaw"
- **Player 2 (Human)**: "eggplant"
- **Player 3 (Human)**: "tangy"
- **Player 4 (Human)**: "servings"
- **Player 5 (Chameleon)**: "frying"
- **Player 6 (Human)**: "tomato"

System: Voting

- **Player 1**: "I think Player 5 is the Chameleon."
- **Player 2**: "I think Player 5 is the Chameleon."
- **Player 3**: "I think Player 5 is the Chameleon."
- **Player 4**: "I think Player 5 is the Chameleon."
- **Player 5 (Chameleon)**: "I think Player 4 is the Chameleon."
- **Player 6**: "I think Player 5 is the Chameleon."

Chameleon Guess: "My guess for the secret word is: Salad"
Outcome: Correct guess! Chameleon escape!

References

Alpaydin, E. (2020). *Introduction to machine learning* (4th ed.). MIT Press.

Arora, R., Basu, A., Mianjy, P., & Mukherjee, A. (2018). *Understanding deep neural networks with rectified linear units.* arXiv:1611.01491 [cs.LG].

Baldi, P., & Sadowski, P. (2013). Understanding dropout. In *Proceedings of the 27th international conference on neural information processing systems - volume 2* (pp. 2814–2822). Red Hook, NY, USA: Curran Associates Inc.

Bjorck, J., Gomes, C., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. In *Proceedings of the 32nd international conference on neural information processing systems* (pp. 7705–7716). Red Hook, NY, USA: Curran Associates Inc.

Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science*, *365*(6456), 885–890.

Chowdhury, G. (2010). *Introduction to modern information retrieval* (3rd ed.). Facet Publishing.

De Deyne, S., Navarro, D. J., Perfors, A., Brysbaert, M., & Storms, G. (2019). The "small world of words" English word association norms for over 12,000 cue words. *Behavior Research Methods*, *51*(3), 987–1006.

Guresen, E., & Kayakutlu, G. (2011). Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science*, *3*, 426–433.

Haarhoff, L. J., Kok, S., & Wilke, D. N. (2013). Numerical strategies to reduce the effect of ill-conditioned correlation matrices and underflow errors in kriging. *Journal of Mechanical Design*, *135*(4), 044502.

Hasan, B. A. S., & Gan, J. Q. (2012). Hangman BCI: An unsupervised adaptive self-paced brain-computer interface for playing games. *Computers in Biology and Medicine*, *42*(5), 598–606.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Jatnika, D., Bijaksana, M. A., & Suryani, A. A. (2019). Word2Vec model analysis for semantic similarities in English words. *Procedia Computer Science*, *157*, 160–167.

Martinez, L., Gimenes, M., & Lambert, E. (2023). Video games and board games: Effects of playing practice on cognition. *PLoS ONE*, *18*(3), e0283654.

Mercier, M., & Lubart, T. (2021). The effects of board games on creative potential. *The Journal of Creative Behavior*, *55*(3), 875–885.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space.* arXiv:1301.3781 [cs.CL].

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning* (pp. 807—814). Madison, WI, USA: Omnipress.

Ornaghi, V., Brockmeier, J., & Gavazzi, I. G. (2011). The role of language games in children's understanding of mental states: A training study. *Journal of Cognition and Development*, *12*(2), 239–259.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics.

Pratiwi, H., Windarto, A., Susliansyah, S., Aria, R., Susilowati, S., Rahayu, L., … Rahadjeng, I. (2020). Sigmoid activation function in selecting the best model of artificial neural networks. *Journal of Physics: Conference Series*, *1471*, 012010.

Richards, M., & Amir, E. (2007). Opponent modeling in Scrabble. In *Proceedings of the 20th international joint conference on artifical intelligence* (pp. 1482—1487). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Shi, G., Zhang, J., Li, H., & Wang, C. (2019). Enhance the performance of deep neural networks via L2 regularization on the input of activations. *Neural Processing Letters*, *50*, 57–75.

Shi, L., Chen, Y., Lin, J., Chen, X., & Dai, G. (2023). A black-box model for predicting difficulty of word puzzle games: a case study of Wordle. *Knowl. Inf. Syst.*, *66*(3), 1729–1750.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-

play. *Science*, *362*(6419), 1140–1144.

Uzair, M., & Jamil, N. (2020). Effects of hidden layers on the efficiency of neural networks. In *2020 IEEE 23rd international multitopic conference (INMIC)* (pp. 1–6).

Webb, G. I., Keogh, E., & Miikkulainen, R. (2010). Naïve bayes. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of machine learning* (pp. 713–714). Boston, MA: Springer US.

Tri Dang is a senior Computer Science and Mathematics double major at DePauw University. His research interests include Data Mining, Machine Learning, and Data Science. He's also open to new aspects of research. He will start his Ph.D. studies in Fall 2025.

Dat Nguyen is a senior at DePauw University double majoring in Computer Science and Mathematics. His main area of interest is Computer Science, in specific Neural Networks, Large Language Models and Natural Language Processing.

Hieu Tran is a Ph.D. student in Computer Science at Purdue University. His research focuses on creating artificial agents that establish long-term connections with individuals, aiming to enhance their well-being, health, and learning.

Brian Howard is an Associate Professor of Computer Science at DePauw University, where he has taught since 2002. He earned his Ph.D. in Computer Science from Stanford University in 1992. His research interests include programming languages, logic, and educational tools to help students learn about functional programming and proofs.

Sutthirut Charoenphon is an Assistant Professor of Mathematical Sciences at DePauw University. She earned her Ph.D. in Applied Mathematics from the University of Memphis in 2020. Her research focuses on partial differential equations, wave equations, and control theory, and the development of mathematical models and computational techniques in these areas.