Codenames AI Report

Xuewei Yan, Yongqing Li, Cameron Shaw

ABSTRACT -FINISHED

Codenames is a popular board game that relies on word association and its ultimate goal is to connect multiple words together with a single clue word. In this paper, we construct a system that incorporates artificial intelligence into the game to allow communication between humans and AI as well as providing the capability of replacing human effort in creating such a clue word. Our project utilized three types of word relationship measurements from Word2Vec, GloVe, and WordNet, to design and understand word relationships used in this game. An AI system is built on each measurement and tested on both AI-AI and AI-Human communication performance. We evaluate the performance with each system's average speed in finishing a game as well as its ability to accurately identify their team words. The AI-AI team performance demonstrates outstanding efficiency for AI to manage this game, and the best performing measurement is able to achieve a 60% accuracy in its communication between AI and Human.

1. INTRODUCTION -TODO

For a long time, people have been using AIs to play games. From Atari games to Starcraft, in the last few decades, artificial intelligence has been making its way into the games humans play to varying levels of success. Games like Othello that have almost entirely been solved by AI, while the ultimate strategy for games like chess and Go just became promising recently.

In this project, we created an AI for the popular board game Codenames. The rules for this board game are simple. There are two teams, each with a spymaster and a fixed number of guesser(s). On the board, there is a five-by-five grid of words, randomly consisting of 9 agents for team A, 8 agents for team B, 7 neutral words, and 1 assassin word. The belonging of each word is only known by the spymasters, and they want to have their guesser(s) identify all their agent words before the other team does. The game progresses in alternating turns, and the spymaster is allowed to give a clue to their guessers each turn. A clue consists of a word which is related to some of their team's agents on the board, followed by a number, which is the number of agents to which the preceding word is related to. For example, a hint of "cloud: 3" is a hint that three of the team's agent words are related to "cloud" in some way. The guessing team is allowed to guess the number of related words plus one additional word each turn. Guesses are made one at a time, and if the guessed word is correct, the next word can be guessed. Each guess eliminates that word from the game board. If the field agents guess a neutral word or an opposing team's

agents, those words are eliminated and the turn ends. If a team guesses the assassin word, the game immediately ends, and the opposing team wins automatically.

Codenames is a game that requires knowledge of how certain words are related to each other. For a human, such knowledge can be different depending on person to person. However, this task can also be executed using a computer. In this report, we investigate some methods that can be used to do this.

2. RELATED WORK -FINISHED

Our project utilizes learned word relationships to simulate the mindset of humans while playing the board game Codenames. In this section, we will discuss relevant word relationships and several existing Codenames AI projects.

2.1 Word Relationship

The entire game functions about word relationships, as the spymaster is trying to find a word that has potential connections to as many words as possible, and the guessers are trying to decode this thought process. Several standardized embeddings have been designed to emphasize inter-word relationships.

[1][2] proposes a vector space word embedding (Word2Vec) that maps each word to a high-dimensional vector. A neural network is trained on the input word corpus so that the vectors of semantically similar words will have higher cosine similarities. Notably, special relationships, such as super-subordinate and part-whole relations, are maintained in the embedding as a constant vector difference between two words. For example, the vector difference between Japan-Tokyo and US-Washington is similar in the Word2Vec embedding.

Noticing the importance of con-occurence frequency in determining the word similarity, [3] proposes a different cosine-similarity-based pipeline, GloVe, that emphasizes this observation. In this project, we will include the word relationship trained from GloVe to compare these two cosine similarity methods.

Another popular word relation corpus is WordNet [4]. Unlike the cosine similarity methods, WordNet looks into the hierarchical closeness of two different words in terms of belonging relationships. For example, “ant” and “spider” (both insects) will have a higher similarity than that of “ant” and “lion”. For this project, we incorporated the Wu-Palmer similarity computed from the WordNet corpus.

2.2 Codenames AI

Several Codenames AI projects exist before our version, and they provide us with strategic guidance as well as some baselines to outperform. For data acquiring, [5] utilized word corpus from Wikipedia to extract the most frequent words. For words about general topics, Wikipedia provides abundant links among words so that the most frequent words can be highly associative to the words provided in Codenames. However, we discovered a preprocessed Wikipedia word list [6] that already does this job for us. We will simply start from there to extract the most frequent words as we need. The author of [7] suggested the potential influence of vocabulary size to the performance of Codenames AI, which encourages us to perform a dictionary size test to validate this idea while finding the optimal vocabulary size for our AI. On the algorithm side, [8] proposed a lower bound for similarity score to be considered for human beings. Multiple AIs who share the same knowledge base can understand the difference between a cosine similarity of 0.1 and that of 0.09. However, word relations of such low similarities become meaningless to humans. Therefore, the author of [8] performed experiments and discovered that 0.45 will be a good threshold to cut off the cosine similarity, so that an AI spymaster will ignore all weak relationships when team up with a human. This information turns out to be useful when we implement the Human-AI mode, and we are expecting different thresholds for different similarity metrics.

3. METHODOLOGY -TODO

3.1 Data

In building the Codenames AI system, we utilized 3 text corpora to obtain different vector embeddings and 3 datasets to obtain the vocabularies for the AI knowledge base. The first corpus we used [9] is a set of preprocessed vector embedding of the top 10,000 most frequent words extracted out of Google's Trillion Word Corpus [10]. We rely on the format of this data for the purpose of building the initial code structure. We would also like to explore other corpora and process our own inputs for word2vec modeling. DBLP is a dataset that contains the titles and abstracts of a large number of computer science papers. This corpus contains enough data but is limited in the range of topics and would require the readers to have domain specific knowledge. Thus we also introduce the Simple Wiki dataset that contains articles from the English Simple Wikipedia. The data is downloaded from the wiki dump and is more general in its choice of topics. This could lead us to a better performing system as more associations can be made with a wider range of topics.

In order to populate our dictionary bases, we first gathered all 400 words used in Codenames from boardgamegeek.com. Then for the AI vocabulary base, we utilized existing projects that

provided the statistics of the most common English words. The first project contains the top 1000 most frequently appearing words, the results came from a n-gram frequency analysis on Google's Trillion Word Corpus. The second dataset presented a similar but longer list of most common words that appeared in the English Wikipedia articles. With the larger file and ranking of word frequencies, we can expand the dictionary size to see its impact on the system performance. We propose vocabulary sizes of 3k, 5k, 10k, 20k, and 30k to test the idea in [7].

To process the DBLP and Simple Wiki corpora, we converted all letters to lowercase, removed punctuations, and tokenized the words. The final format of the corpus is a 2D list that contains the word tokens for each individual article. Now the data is processed for training. We utilized the Word2Vec model from the gensim package to train the vector embedding for each word in our Codename and AI vocabulary bases. This model takes in a correctly formatted text corpus and outputs a vector in a 300 dimensional space for each word. The vector embeddings for the words used in the both vocabulary bases are then extracted to compute the similarities.

We used 3 approaches, cosine, path, and Wu-Palmer, to measure the similarities between different words. Cosine similarity is obtained by calculating the matrix multiplication of the AI vocabulary and the transpose of the Codename vocabulary. Path and Wu-Palmer similarities comes from using pre-trained wordnet models from the nltk package. In the end we outputted a total of 33 sets of word embeddings, 3 similarity calculations for each of the pre-trained text corpus and the 5 dictionaries with varying sizes for the DBLP and Simple Wiki corpus.

3.2 Algorithms

Algorithm for Spymaster

Our Codenames AI optimizes each step of its action based on the current board information *B,* the vocabulary base *W* it has, and its knowledge about word similarities in terms of a matrix *M*. When playing as a spymaster, it will go through all of its vocabulary and try to optimize the clue through the following criteria: 1) Connect to the greatest number of ally words on the current board; 2) Is away from other words in terms of similarities as much as possible.

Algorithm 1: Naive Codenames Clue Nomination

Input: Available words on board *B*, set of ally words *A*, vocabulary base *W*, similarity matrix *M*.
Output: The optimal word as the clue *w**.

$smax \leftarrow 0$
*w** ← None, *c** ← 0
for *w* **in** *W* **do**
    $U \leftarrow \{x | x \in B \text{ and } x \notin A\}$ *// Set of non-ally words still on board*
    $l \leftarrow max\{M(w, u) | u \in U\}$

$c \leftarrow 0$

$m \leftarrow 1$ // *Record the smallest similarity that is greater than l*

for v **in** $A \cap B$ **do**

if $M(w, v) > l$ **then**

$c \leftarrow c + 1$

if $M(w, v) < m$ **then**

$m \leftarrow M(w, v)$

$s \leftarrow c + m - l$ // *Adjusted count as if* ***w*** *is proposed as the clue. The value of* ***s*** *is dominated by* ***c****.*

if $s > smax$ **then**

$w^* \leftarrow w, c^* \leftarrow c, smax \leftarrow s$

return w^* , c*

To achieve this, we design an algorithm (Algorithm 1) that computes a score *S(w; B, W, M)* for each vocabulary word *w*, and finally pick the word with the highest score as the clue proposed. For each *w*, we first extract its similarity with all the existing words on the board. We compute a lower bound l_w as the maximum of the similarities among all non-ally words (i.e. opponents' words + neutral words + assassin word). Then, we can have a set S_w composed of all ally words that have higher similarities than l_w. Assuming our guesser shares the exact same knowledge for the similarities with the spymaster, it is guaranteed that the guesser will guess all words in S_w before touching any of the non-ally words if provided with the clue *w*. Having this in mind, the word *w** will be the optimal clue if the size of S_{w*} is the largest. To break ties, we implemented a safety weight that slightly prefers the word with a larger gap between l_w and similarities for S_w.

The naive version of the clue nomination algorithm is strengthened in the following ways: 1) When computing the maximum similarity of non-ally words for the lower bound l_w, we assign fixed weights (> 1) to words that we do not want the guesser AI to touch. Explicitly, the assassin word will get the biggest weight because touching it will make the team lose automatically; the enemy's words have a higher weight than the neutral words for a similar reason. 2) When playing with a human teammate, the lower bound will become $min(l, \theta)$ where θ is the threshold for the connection to be human-interpretable, inspired by the observation in [7]. Qualitative and quantitative methods are mixed to give suggestions to the threshold θ for each system, and details will be explained in Section _____.

Algorithm for Guesser

After receiving the clue word *w* and the number of words related to the clue *c*, the guesser will make a sequence of guesses based on the same idea. It ranks all words on the board based on their similarity with *w*, and makes a guess on the word with the toppest rank. If succeeded, it proceeds to the second, and the third, … The procedure ends until a non-ally word is touched, or the guess count reaches the limit.

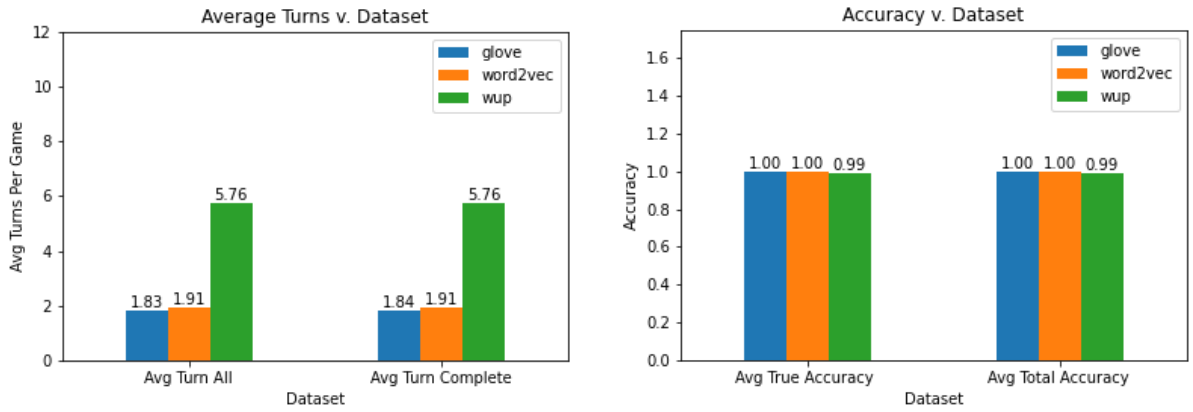
However, one should notice that, assuming both the spymaster and the guesser are AI players who share the same knowledge base, there will be no chance for the guesser to get any guess wrong, because the top k associated words are guaranteed to be ally words by Algorithm 1. This makes the game played by pure AI teams finish super fast, as observed in Section _____. To simulate the fact that different humans, although sharing the same commonsense of the world, may have slightly different perceptions towards the strength of word similarities, we add a noise matrix to the similarity matrix of the guesser. Such a matrix modifies the value of each similarity, with each point of the noise matrix sampled from a normal distribution with a small standard deviation. By increasing this standard deviation, we are able to simulate two teammates with more different mindsets and less so-called "tacit understanding". Details of this experiment can be found in Section _____.

4. EXPERIMENTS AND RESULTS

In this section, we test for the performance of the three word relationships under the settings of an AI-AI team and an AI-Human team. The performance will be assessed with different metrics and standards to reflect different expectations of the relationships under these two team settings. Ultimately, we want to find the top performing combinations that can be deployed to optimize both types of communications. To test for the best performance, hyperparameters have already been tuned beforehand for each word relationship. Details for these tuning processes can be found in *Section 5*.

4.1 AI-AI Communication

For AI-AI communication, we primarily test on the accuracy of guessing and average number of turns for the AI to finish the games. The word relationship datasets used in this test are three similarity matrices – Word2Vec, Glove, and Wu-Palmer, correspondingly – with 30,000 vocabulary each for the AI to use. The statistics are obtained by simulating 100 random games per word relationship, with AI taking all 4 roles: 2 spymasters and 2 guessers. To avoid confounding factors, we controlled the experiments by using the same set of random seeds for each set of word relationships.



(a) (b)

Figure 1. Three datasets are each tested with 100 simulations in the average number of turns taken to finish a game and its ability to accurately choose the intended words from the spymaster. GloVe and word2vec datasets perform better than that of the wup dataset.

As observed in figure 2a, the GloVe and Word2Vec dataset performed equally well than that of the Wu-Palmer dataset. _____ Our hypothesis of high values in game accuracy from AI-AI communication is also tested with the final datasets. Figure 2b visualized both the true and total accuracy for each dataset and we can see an almost perfect performance for AI guessers to correctly choose the intended words from the spymaster. This is again due to the sharing of knowledge bases between the AI players. Although not the main focus, we also observed a slightly better performance in the GloVe and word2vec datasets.

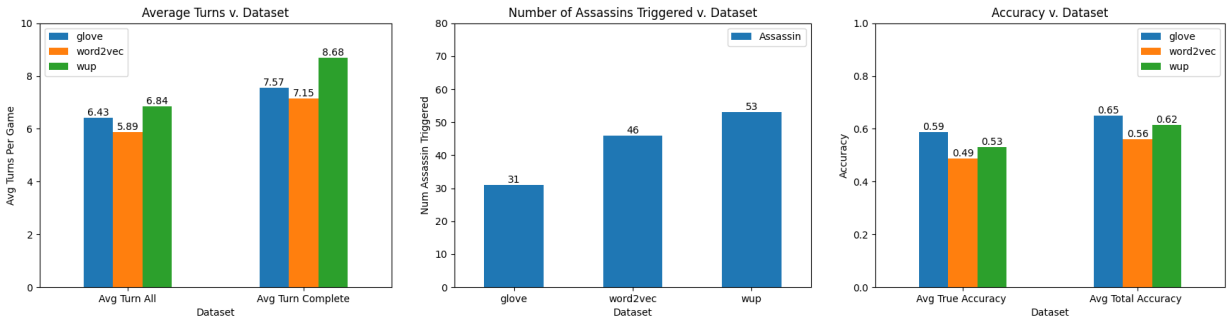
4.2 AI-Human Communication

For this section of the experiment, we use the same experimental set up as AI-AI testing, with the exception of substituting the AI guesser with a human player and testing with a smaller group of datasets. Similar to above, we will take into consideration the average number of turns taken to finish the game. Now that the players are using different knowledge bases, we will also examine their interactions in terms of how accurately the human guesser can uncover the AI intended words.

For each of the simulations, we record the same data as the above experiment as well as statistics regarding the accuracy per game. Every round, the AI spymaster will generate a clue word and have a set of target words that it intends to connect with. We record these intended words along with the guesses made by the human guesser to calculate the accuracies for each game. We measure 2 types of accuracies in this section. The first one, true accuracy, being the number of team words that are correctly guessed as well as intended by the AI. The second one, total

accuracy, is calculated as the number of team words that are correctly guessed and these do not have to be in the set of words intended by the AI spymaster.

We carry out the same experiments as above and replace the role of AI guessers with humans for both teams. With the introduction of human efforts, the average turns taken for a game to finish has increased in 5 out of the 6 experiment settings (figure 2a). This behavior is expected, however, as there exists a gap between human and AI communication. Similar to AI-AI communication, the GloVe and word2vec datasets had better performance. We then compare each datasets' impact on triggering the assassin word, which would cause an immediate termination of the game. Figure 2b presents the GloVe dataset as having the most number of complete games in a simulation with 100 games. Finally we analyze the accuracies of each dataset in figure 2c. GloVe again stood out with the highest accuracies amongst the 3 datasets. It is interesting to see that even though the word2vec dataset finished the games faster, its accuracy in performance was the worst. As both high accuracy and less turns are needed to succeed in the game, the tradeoffs between the two will need to be balanced by the players based on the preferred goal. In this section we conclude that the GloVe dataset is most compatible with human players given its high accuracy and low game termination.



(a) (b) (c)

Figure 2. Three datasets are each tested with 100 simulations in the average number of turns taken to finish a game, number of assassin words triggered, and its ability to accurately choose the intended words from the spymaster. The GloVe dataset performed best when human players are introduced to the game.

5. SIDE EXPERIMENTS AND DISCUSSIONS

5.1 Vocabulary Size

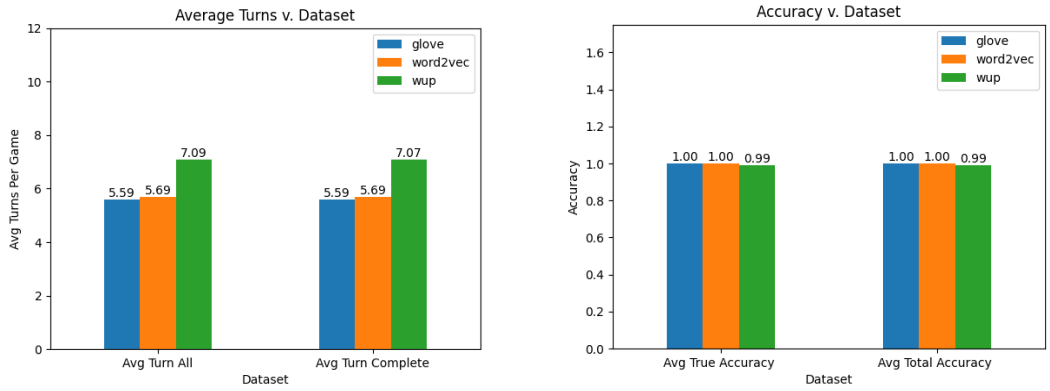
5.2 Human Interpretable Threshold

As discussed in [8], although the AI-AI team can take advantage of the full similarity matrix when proposing clues, the word relationship with a relatively low similarity score is meaningless

to a human being. As a result, when the AI spymaster is teaming up with a human, proposing clues with high number of connections but low similarity scores will have much stronger side effects than benefits. Thus, for the betterment of the AI-Human team performance, it is effective to remove all the word relationship that have a similarity score lower than a certain threshold. In practice, we set the similarity value of these cases to zero so that the AI won't see any connection of the pair of words.

The straightforward way to determine the threshold is to manually evaluate percentage of meaningful connections among all pairs of words. To do this, we randomly sampled 10 words from the codenames word list, and sort all the vocabulary words by their similarity score with the codenames word. Some examples of this process can be found in Appendix B. By observation, the top 30 words are super strong connections that a human can identify with no efforts. However, having only 30 connections per word makes the final matrix very sparse, hence producing no clues with connections higher than one. We lower the expectation and conclude that the top 80-120 words contain some meaningful connection to the target word. This is our secondary threshold that is acceptable but not satisfying.

In the end, we chose the threshold to be the 96.5th percentile of the entire matrix, which is on average 80 words per target word. We did another round of 100 game simulation of AI-AI testing on this sparse version of dataset, where the human threshold restriction is enforced. The result is shown in Figure 3. The average number of turns is roughly the same with the AI-Human testing results, which indicates that our threshold restriction pushes the performance of AI closer to that of a Human.



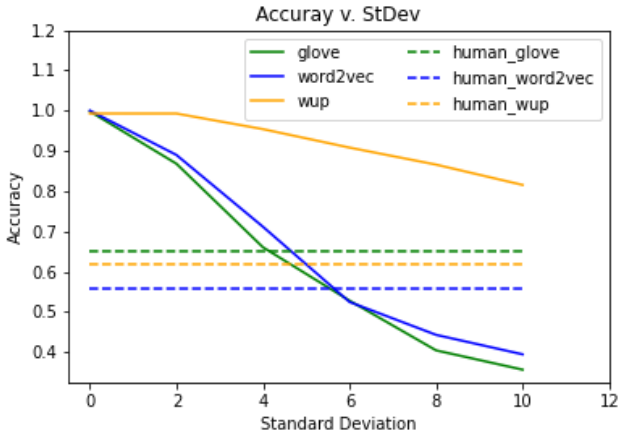
(a) (b)

Figure 3. Average number of turns and accuracy for AI-AI testing (N=100) with human threshold restriction enforced.

5.3 Level of Human-AI Misinformation

A significant advantage of the AI-AI team is that both the spymaster and the guesser use the same word relationship matrix to make decisions. Hence, their knowledge base is completely the same. On the other hand, this is the main reason for this board game to be entertaining for human but trivial for AI.

To simulate the gameplay of Human, i.e. different individuals have different knowledge base for the word relationship, we add a confusion matrix as noise to the matrix of the guesser. Each element of the confusion matrix is sampled from an i.i.d. Gaussian distribution with mean 0 and standard deviation σ. The higher the value of σ is, the more different the knowledge bases of AI spymaster and AI guesser will be. We would expect an increase in the average number of turns and a decrease in the accuracy as σ increases. To see how much of a standard deviation can make AI performs similar to a Human in terms of these statistics, we did a set of experiments with different sets of thresholds.



6. CONCLUSIONS AND LIMITATIONS

References

[1] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781

[2] Mikolov, Tomas (2013). "Distributed representations of words and phrases and their compositionality". Advances in Neural Information Processing Systems. arXiv:1310.4546

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
[4] Princeton University "About WordNet." WordNet. Princeton University. 2010.
[5] Kirkby, David (2017). “CodeNames AI”. Github repository,
https://github.com/dkirkby/CodeNames.
[6] Semenov, Ilya. “wikipedia-word-frequency”. GitHub repository,
https://github.com/IlyaSemenov/wikipedia-word-frequency/tree/master/results
[7] Shafir Michael (2017). “Codenames AI”. Github repository,
https://github.com/mshafir/codenames-ai.
[8] Pbatch (2020). “Codenames”. GitHub repository, https://github.com/Pbatch/Codenames.
[9] Kaufman, Josh. “google-10000-english”. GitHub repository,
https://github.com/first20hours/google-10000-english.
[10] Jean-Baptiste Michel*, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, William Brockman, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden*. Quantitative Analysis of Culture Using Millions of Digitized Books. Science (Published online ahead of print: 12/16/2010)

Appendix

Appendix A. Project Proposal

Codenames is a popular word association game that involves 2 teams, 4 roles, and 25 word cards. The cards are laid out in a 5x5 display and each team gets randomly assigned 8 or 9 words. The rest of the cards are all neutral except for 1 assassin card. The teams compete by designating a spymaster, who knows the identity of all the cards, to give a one-word clue along with a number for the guessers, the rest of the team, to identify which cards belong to their team. The two teams will take turns until either one team correctly guesses all their words, or victory is declared to the opponents if one team revealed the assassin card.

We are interested in developing an AI system that could replace the roles of the spymaster and/or guesser for the game Codenames. This will allow for humans to team up or compete with AI and we hope to discover interesting behaviors during the process. As the game involves associations between different words, we will investigate the relationship for all word choices provided by the game. Data will be obtained from the website boardgamegeek.com which consists of the 400 words used for the game. In order to accomplish this, we will utilize Word2Vec models and transform the words in the game and a dictionary consisting of the most common 10,000 English

words into vector embeddings. Then we can produce good word clues by using cosine similarities as words that are closely associated should have a higher similarity score.