Codenames Clues: Princeton Edition
TRA 301 Final Project

Nora Graves
Julia Ying

May 6, 2025

Contents

1 Introduction & Motivation

Games are crucial to human development, quite literally helping the brain grow [15]. In fact, play is so important that even the United Nations (UN) considers it a human right for children [41]. The benefits of games extend past childhood and into adulthood, and time spent playing video games positively correlates with positive mental health [19]. In the past few years, word games in particular have seen a significant increase, ranking as the fifth most popular genre of games, played by over 20% of mobile gamers [12]. These word games have significant benefits, and can even increase cognitive abilities like memory access and vocabulary [35]. Although many of these games, such as Wordle, are single-player, there are also multi-player games, such as Scrabble, Bananagrams, and Codenames. In this project, we focus on Codenames for a variety of reasons. First, Codenames is extremely popular, and has won a variety of awards, including Spiel des Jahres, a prestigious German board game award [16]. Second, it relies on a simple interface and set of rules, but requires complex strategy, a combination that renders it an ideal application of machine learning (ML) and natural language processing (NLP). In fact, researchers have proposed Codenames as a benchmark task for large language models (LLMs) [34]. Finally, Codenames offers a valuable opportunity for customization, by adding unique words into the vocabulary.

The Codenames game setup and rules are as follows [11]: Players are split into two equal teams, one blue team and one red team, with one person on each team designated as the "spymaster." Cards with words on them are laid out into a 5 x 5 grid such that everyone can see them. Both spymasters also get an additional key card that reveals which words belong to which teams (blue or red), which words are neutral, and which word is the "assassin." The spymasters alternate turns. During their turn, the spymaster gives a clue word in addition to the number of cards that the clue relates to. Their teammates have up to that number of guesses to guess the words that fit that clue, one word at a time, and then each card is revealed (blue, red, neutral, or assassin). If the teammates guess incorrectly, their turn ends. If the assassin word is guessed, the team loses automatically and the game ends. Otherwise, the first team to guess all of their

words wins the game.

Our project integrates Princeton-related slang into the Codenames gameplay. Out of the 25 words placed on the board, 5 are chosen to be Princeton-related. We then aim to create a machine translation model that is able to generate successful clues given the words and each word's identity.

Dialects and jargon, like games, are incredibly beneficial for humans. Socially, they help build group identity, and contribute to bonding and trust [17, 26]. People can consistently identify various group memberships based on vocabulary alone [33]. On college campuses, creating a sense of identity is especially important [20], which is why many campuses develop specific jargon. Linguists have studied this phenomenon for centuries, noting words like "dorm," "frat," "prof," and more [10]. Princeton, too, has its own unique vocabulary. Words like "bicker," "prox," and "COS" differ from similar words at other campuses [6]. Given the importance of unique dialects and our own position as Princeton University students, we decided to develop agents capable of generating Codenames clues from not only standard Codenames words, but also Princeton-specific words. These agents will not only offer insight into various clue generation strategies, but also help individuals learn to play Codenames, as well as practice Princeton jargon. Our project contributes to the field in the following ways:

1. Add new vocabulary to pre-existing word embeddings, using fine-tuning methods.
2. Customize Codenames with specialized language, in this case, Princeton slang.
3. Train an agent to create clues for this customized Codenames, using word embeddings.

2 Related Work

This project stems from the field of Natural Language Processing (NLP). In particular, we draw on research on word embeddings, as well as prior attempts to develop agents capable of playing the game Codenames.

2.1 Word Embeddings

Word embeddings, as the input and output of the neural net itself, are crucial to LLMs. Research into complex, trainable, and analyzable embeddings began with the introduction of Word2Vec in 2013 [25], a system which remains popular today, even as word embeddings have changed slightly since the establishment of transformer-based LLM architectures [14, 29]. Over the years, other methods for training, visualizing and analyzing word embeddings have been introduced [28, 4].

By quantifying the complexities of natural language, word embeddings not only make the neural net within LLMs possible, but also allow for the application of various vector operations. Perhaps the most common operation is cosine similarity, which many researchers have used to analyze the relationships between various pairs of words [28, 36, 27], but other common operations on word embeddings include linear transformations [2] and low-rank approximations [28, 9]. In particular, this project builds on vector addition and cosine similarity; together, these two methods help identify similar words, and create a clue that relates to all words chosen.

Word embeddings can also be fine-tuned, though this has not been a particularly popular research topic; most projects choose to fine-tune at a higher level, modifying the final layers of the LLM, rather than its core word embeddings [38]. Furthermore, strategies tend to fine-tune the existing vocabulary, not add new words [24, 40], as this project aims to do. Nonetheless, fine-tuning word embeddings proves successful, improving performance on tasks like emotion recognition [42] and essay scoring [13]. Notably, fine-tuning can even be successful with a fairly small corpus, although sometimes a corpus that is far too small can have adverse effects [22]. In this project, we aim to fine-tune word embeddings for the specific purpose of adding new vocabulary, in order to create Princeton-specific embeddings.

2.2 Codenames Strategy

With the growth of AI, particularly its NLP applications, developing Codenames agents has become an increasingly popular challenge, as it requires both cooperative abilities

as well as complex language understanding. Researchers have attempted a variety of methods. Transformer-based methods have proven more successful than the baseline Codenames bots (built using Word2Vec and GloVe) when playing as the spymaster, but performed worse as a guesser [18]. Often, when the same bots play each other, they perform well, but perform significantly worse when playing against bots with different implementations [21]. In this project, instead of bot–bot interactions, we focus specifically on bots who play with humans, following the methods of other researchers [5]. Besides word embeddings, other Codenames strategies include strategic reasoning based on *all* prior clues [3], co-occurance calculations [5], and methods of adding noise to improve generalizability [1]. In this project, we use word embeddings [39, 8, 23], and expand on the field by adding new words to the vocabulary, creating a Codenames game which includes Princeton-specific vocabulary.

3 Approach

In this section, we explain our general approach, and the methods we considered in our project. Drawing from the related work, we decided on word embeddings as the most flexible and efficient Codenames clue generation strategy. Therefore, our complete approach consists of the following steps: creating a Princeton-themed training corpus, fine-tuning a word embedding model, and building a Codenames simulator.

3.1 Domain-Specific Corpus

In order to create word embeddings for Princeton-related words, we had to utilize a corpus of Princeton-related text. Since there was none that we knew available, we decided to curate one ourselves. We both manually and automatically scraped data from large text sources such as Wikipedia and *The Daily Princetonian*. In addition, we developed a list of Princeton slang/jargon to be the words added to the Codenames board, and we later iteratively ensured that the scraped corpus had large enough counts of these words.

3.2 Word Embedding Models

We created the word embedding model using Word2Vec, because of its ease of use. We initially considered training a word embedding model on just the Princeton corpus, but realized that the original Codenames game words would be missing or poorly represented in the model. We then tried many combinations of integrating an existing word embedding model with our vocab, with varying degrees of success. We initially tried downloading Google News, trained on 100 billion words from Google News, but due to compute and time restraints this ended up being too large of a model. We pivoted to a model created by Global Vectors for Word Representation (GloVe) representing corpuses from Wikipedia 2014 and Gigaword 5, about 6 billion tokens [28]. Using this model, we eventually built a vocabulary out of our Princeton corpus and the Codenames words. We then added the original word embeddings from the model for any word already in the model and trained it on our scraped Princeton corpus.

We used this model to generate Codenames clues for both the user and the opponent. We did so by using functions from Word2Vec such as `most_similar()` and `similarity()`. Since each word is treated as a vector, similarity in Word2Vec is computed using cosine similarity, where values closer to 1 mean more similar and -1 mean less [7].

3.3 Codenames Simulator

Finally, we created a Codenames simulator using the model described above as a clue generator. We initially implemented this in a Google Colab notebook, but as complexity grew, shifted to a full-stack website with a Flask backend and a React and Vite frontend. On the website, the 5 x 5 board is displayed, and the user is provided the clue generated by the model. They are then able to input guesses one by one (including the extra guess allotted by the Codenames rules). The team their guesses belong to are shown on the board after each guess. We also implemented an opponent that plays automatically, basing their guesses also on word similarity with some fixed error.

4 Implementation

4.1 Princeton-related Data Curation

First, we scraped some Princeton-related Wikipedia articles, using the `wikipedia-api` package. From this, we added detailed information about Princeton itself, as well as important buildings such as Nassau Hall, the eating clubs, and residential colleges, to our training corpus.

In addition, we scraped the majority of our Princeton-related training corpus from *The Daily Princetonian* (nicknamed the "Prince") website, Princeton's student-run daily newspaper, established in 1876 [30]. For this we used `BeautifulSoup`, a Python package for parsing HTML. We began by scraping articles from each section of the Prince, yielding the ten most recent articles from each. We did this on three separate occasions, spread across three weeks, resulting in a wide sample of recent Prince articles.

We curated a list of 76 Princeton words that we thought would be fitting for the Codenames game board. We selected these words based on popularity and uniqueness, and it includes, for example, "bicker," "Rocky," "Tiger Inn," "PSafe," "late meal," and more (see Appendix A.2). Given this list, we analyzed our scraped corpus, identifying the words appearing least frequently thus far, and manually found articles with those words to add to the corpus. For example, residential college names (Rocky, Mathey, etc.) appeared frequently in Prince articles about room draw.[1] Finally, we added a list of definitions curated by *The Daily Princetonian* specifically for first-years, nicknamed the Frosh Dictionary [31]. Prior work suggests that dictionaries, although not typically used to train embeddings, can be quite successful [37]; in this project, they served as a valuable supplement to our corpus.

We were also planning on scraping a popular Princeton-student-run Instagram account affiliated with Barstool Sports, a media company known for its sports and pop culture content, called @barstoolprinceton. However, we were unable to bypass Instagram's strict

[1] "It all comes down to this: 2025 upperclass draw," "University overhauls floor plans ahead of room draw," "New pre-draw guidelines reflect a much-needed conversation about community care on campus," etc.

anti-scraping measures.

Ultimately, we were able to create a corpus from these variety of sources that consisted of 629,845 words, and contained at least 8 instances of each item in our list of Princeton-specific words for the Codenames gameboard.

In order to use this corpus, we also needed to clean it. After converting everything to lowercase, we first identified synonyms for Princeton terms (e.g. "Tiger Inn" and "TI" are equivalent) and standardized them, which further increased the representation of relevant words. Next, we converted all multi-word items (i.e. "Lake Carnegie" and "Late Meal") into a single token by replacing spaces with underscores. Finally, we used the `nltk` package to finish cleaning and preparing the dataset, then it was ready for fine-tuning the word embedding model.

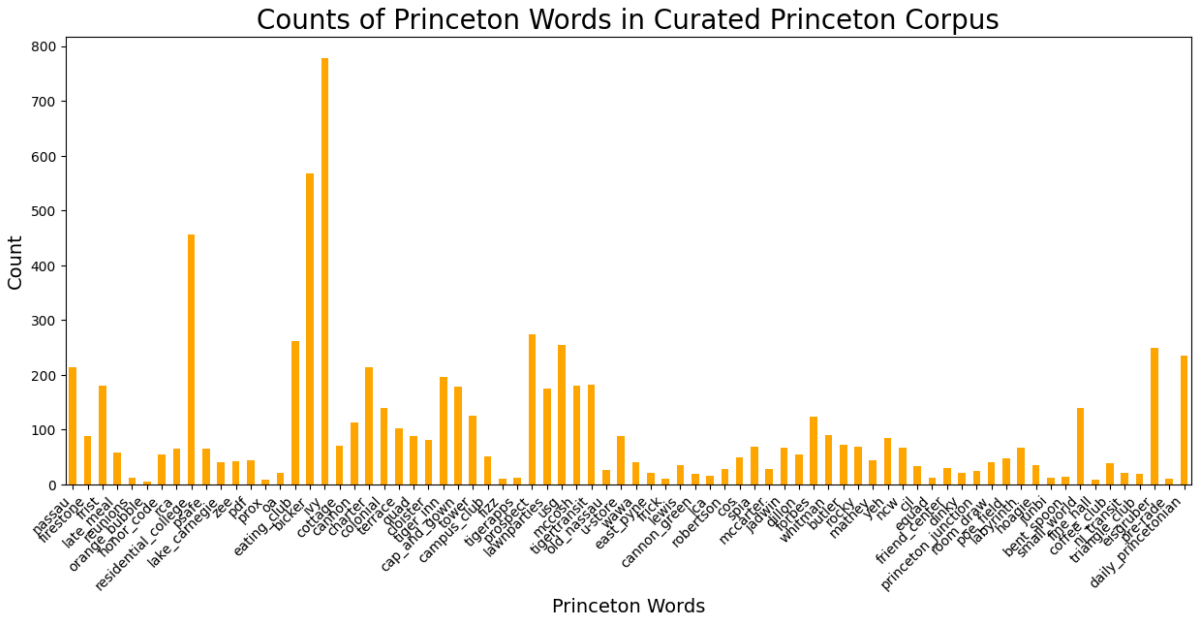


Figure 1: Counts of Princeton Codenames Words in Corpus.

4.2 Word Embedding Model Results

We attempted fine-tuning the word-embeddings with our corpus twice. First, we used a pre-trained GloVe model with embeddings of size 100,[2] and used the hyperparameter that a new word must appear at least 5 times in the corpus to receive an embedding. This

[2]GloVe Wiki Gigaword 100, available at `https://nlp.stanford.edu/projects/glove/`

model was unsuccessful, which we determined through manual inspection of the newly added Princeton words. The top 10 most similar words to "Fine Hall," for example, included insignificant, infrequent, and unrelated words like "norbrook," "mastnak," and "slakey." With a brief survey of Princeton students, we determined that only one of the ten most similar words ("PPPL," a.k.a. the Princeton Plasma Physics Lab) had any meaningful connection with Fine Hall; thus, we determined that these word embeddings would not lead to successful clues in a Princeton themed Codenames game.

Next, we used another pre-trained GloVe model, with embeddings of size 300,[3] and modified our hyperparameter such that a word must appear at least 8 times in the fine-tuning corpus to be added to the model (based on our corpus). This model was far more successful, as it helped eliminate infrequent words such as "norbrook," creating more relevant connections between words. In this model, there was a higher ratio of relevant words; "Fine Hall," for example, was similar to "down-campus," "B.S.E.," and "serveries."

As we were implementing the word embedding model, we realized that while we were able to standardize and include the multi-word Princeton words such as "late meal," there was no easy way to do the same for the multi-word Codenames words; the pre-trained word embeddings model we used to supplement our Princeton vocabulary did not recognize the multi-word Codenames words like "ice cream" as a valid input. As there were only four multi-words on the list, we chose to remove them (see Appendix A.1 for a full list of Codenames words, including the four we removed).

The final modification we made to our model was recalculating the frequency of each word in the vocab. This will be necessary for creating clues, because ideally clues should not be extremely obscure words. In the fine-tuned model, the word frequencies were calculated from the training corpus, rather than English use more generally. Therefore, we re-sorted the words using a dataset of English words and their frequency [32]. Additionally, we manually increased the frequency of Princeton specific words (see Appendix A.2), to ensure that restricting clues to frequent words only would not negatively affect

[3]GloVe Wiki Gigaword 300, available at `https://nlp.stanford.edu/projects/glove/`

any Princeton-related clues. See Appendix A.3 for a table of the Princeton words we chose to add to Codenames along with the top 10 most similar words our model outputs and its similarity score.

4.3 Clue Scoring & Ranking

Our clue generation consists of two steps. First, the agent must determine its target words from the remaining words on the board, for which it will produce a clue. Second, it must create a clue, ideally avoiding opponent's words on the board.

Choosing Target Words Initially, we designed the agent to choose its first target word randomly, then identified all other team-words with a high similarity to that target word. Although this strategy successfully created sets of target words when possible, the random initialization led to a high likelihood that there would be only a single target word, which is an inefficient and non-ideal Codenames strategy. Therefore, we modified our algorithm: First, search for the pair of team-words with the highest similarity. If this similarity is above the threshold[4], then iterate through all the team-words again, adding any additional words which are similar to that pair. This creates a target set containing two or more similar words. If, however, the selected pair has similarity below the threshold, then no safe set of target words is possible, so the agent chooses a single target word with the lowest similarity to all other team-words.

Generating a Clue Word For clue generation, we tested a variety of strategies, using the `most_similar()` function implemented in the `Word2Vec` package in Python. This function takes two main arguments: a list of positive words (i.e. target words), negative words (i.e. words from which the output should be different), and a numeric value by which to restrict the vocab. It outputs a list of tokens, and their similarity to the target words. In all variations, the target words chosen as described above served as the positive

[4]Through trial and error, we determined that a similarity threshold of 0.25 generated pairs with meaningful similarity, without creating incredibly long lists of target words. After all, human players tend to struggle with clues intended for 4 or more words.

words argument. From the list of similar words produced, we select the first word that is not numeric, and satisfies Codenames rules about legal clues.

Initially, we added all non team-words (i.e. all remaining words *not* belonging to our team) to the negative words argument. This strategy, however, was unsuccessful: it almost always produced extremely obscure words. Even a simple evaluation by inspection made it quite clear that these words would fail as clues.

Next, we limited the negative words argument to only potentially problematic words, once again relying on the threshold hyperparameter. In this case, if a words similarity to the target words was below the threshold, than a clue is unlikely to accidentally reference it; therefore, the `most_similar()` function does not need to purposefully avoid similarity with that word. This method led to a far smaller set of negative words, so `most_similar()` produced a lower ratio of obscure words. Nonetheless, the top clue generated was often still obscure, even though other potential clues were not.

Finally, we restricted the vocab to more frequent words, which prevented `most_similar()` from generating obscure words. This method finally led to reasonable clues. For example, the words "deck" and "ship" resulted in the clue "boat, 2." Because this clue generation method no longer had obvious errors, we proceeded to implement the Codenames simulator.

4.4 Codenames Simulator Implementation

We were able to implement our Codenames simulator as a full-stack website at `https://princetoncodenames.onrender.com/`[5]. This website was built with a Flask backend (given our code was written initially in Google Colab in Python), a React and Vite frontend, and deployed on Render. The user is designated as the blue team with 9 total cards, and the computer as the red team with 8 total cards. There are also 7 bystander cards and 1 assassin. The user is able to see the board of 25 cards, 5 of which are Princeton words, and the clue given. On their turn, they can input their guesses one by one and see them revealed. After each user's turn, they are able to let the computer play,

[5]Note that the website may take 1–2 minutes to load.

described below.

Opponent Game Play For our opponent, the "computer," we generated the clue in the same way as for the user. However, we then calculated the similarity scores of reach word on the board with the clue, and with some accuracy, guesses the possibilities in order, otherwise guess randomly. We set this accuracy to be 90%.

5 Results

For this project, we completed two types of evaluations. First, we ran repeated clue generation on randomized boards, and collected quantitative data. We also performed user evaluations on Princeton students who were familiar with both Codenames and the Princeton vocabulary that we were exploring in this project.

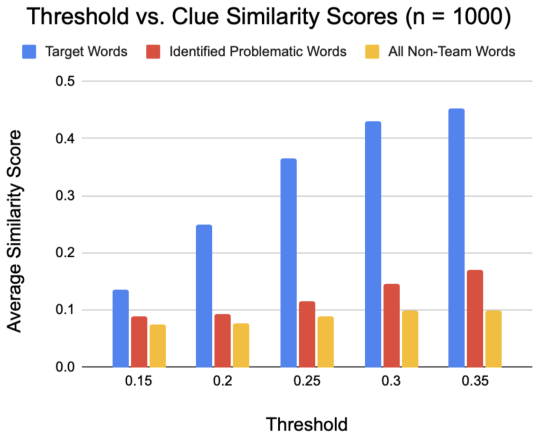
5.1 Quantitative Evaluation

In our quantitative analysis, without human players, we evaluated in two ways: first, across a range of threshold options, and second, with the specific threshold we selected (0.25). For both, we ran 1,000 trials in which we initialized a random board, and then generated a clue from that board. Ideally, we would like to perform analysis beyond the first clue in each game as well. Unfortunately, doing so would require human players to take their turn, continuing the game. Therefore, in the interest of efficiency, we performed our large-scale quantitative analysis separately from our user studies.

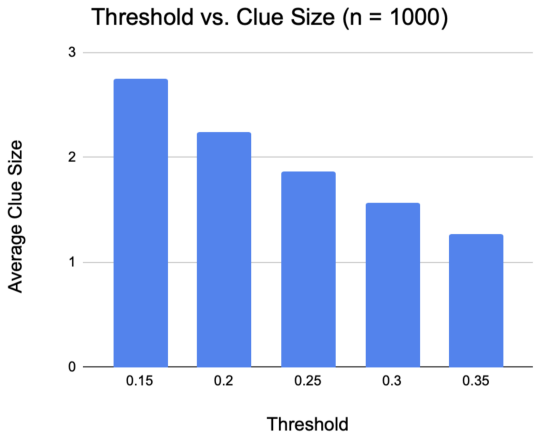
Similarity Threshold When developing the Codenames simulator, we left `threshold` as a modifiable variable. With a lower threshold, the clue generator would select less similar words as target words, while a higher threshold would force it to select only very related words as target words. We used the same `threshold` variable to identify potentially problematic words on the board (words not for our team that are very similar to the target words). Unsurprisingly, the threshold variable resulted in a tradeoff (see Figure 2).

On one hand, a higher threshold means that the clue will have a much higher similarity to the target words. This is because the target words will be quite similar to each other, increasing the likelihood that an additional word (the clue) will be very similar to all of the targets. Unfortunately, we do also see that the similarity between the clue and the problematic target words increases as the threshold increases. Rather than signifying a flawed algorithm, this simply reflects the fact that, as the threshold increases, words identified as problematic must be increasingly similar to the target words; it therefore becomes increasingly difficult to find a clue which is not similar to the problematic words.

On the other hand, as threshold increases, the number of target words (i.e. the clue size) decreases. This is not ideal, because clues with more target words, allow players on a team to guess more words at once, allowing them to more quickly identify all of the words for their team, leading to a win.



(a) As the required similarity threshold increases, the similarity between the clue and the target words increases.



(b) As the threshold increases, the average number of target words selected decreases.

Figure 2: Threshold tradeoffs.

Based on this tradeoff, along with some qualitative feedback from human players, we ultimately settled on a threshold of 0.25, which resulted in an average clue size of 1.862 (median = 2, standard deviation = 0.5012). Furthermore, this threshold produced clues with a high similarity to target words (mean = 0.3663, median = 0.3872, standard deviation = 0.1509), but a low similarity to all non-team words on the board (mean = 0.0893, median = 0.0862, standard deviation = 0.0257) (see Figure 3). Qualitatively,

the threshold of 0.25 resulted in reasonable sets of target words and meaningful clues.

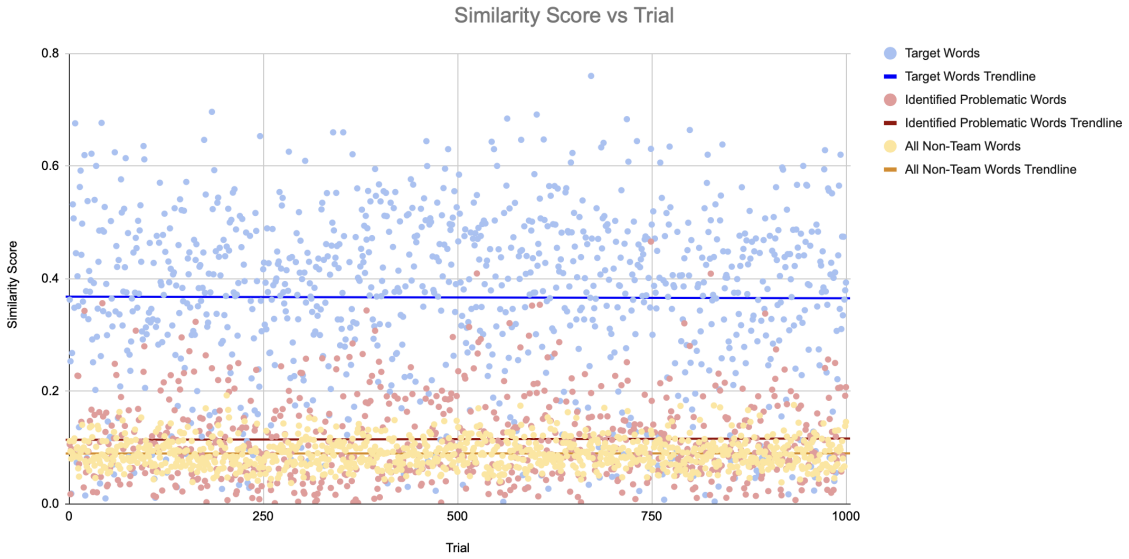


Figure 3: Similarity scores between the generated clue and various sets of words on the Codenames board across 1,000 trials. The clues are consistently far more similar to target than non-target words.

5.2 User Studies

In our final user evaluation, we conducted surveys on Princeton students to collect feedback on our model. We asked them to play multiple Codenames games on the website we built, where they can play as a guesser with our agent as their Spymaster teammate, and bots as their opponents.[6]

We tested our Spymaster agent by asking 5 fellow Princeton students with some familiarity with Codenames to play our game to completion 3-5 times, and collecting feedback on good clues, bad clues, and statistics on the number of correct guesses made by the human player, the rate of winning, etc. All together, the users played 18 games. 15/18 of the games the user was able to win, with 7/15 of those games which were won due to the user guessing all their cards (as opposed to computer winning or someone picking assassin, which may suggest that we can actually increase the accuracy of the opponent). Out of the 7 games where the user was able to guess all their cards, the

[6]Because bots show unnaturally high success when teamed up with bots with the same implementation [21], we implemented the bot opponent team with some random error built-in.

average number of rounds it took for the game to end was close to 9. From experience, this seems like a reasonable number of rounds for the game to last with clues given with 1-2 words and some wrong answers.

It seems that most clues were not Princeton-related, which makes sense given only 5 out of the 25 total cards were Princeton-related, meaning that the user's team would only have potentially 1–2. Some good clues for the regular Codenames words that were noted were: CHINA → BEIJING, CAR → ENGINE, INNING → PLATE, TAIL → SPINE, MOUNT → OLYMPUS. In particular, APPLE → TABLET was interesting because it shows that the model is able to pick up modern culture such as companies. Some Princeton-related ones were TIGERJUNCTION → TIGERAPPS, which references one of TigerApps' most popular applications, BARISTAS → SMALL WORLD, BENT SPOON, which was able to get two dessert/coffee places at Nassau together, and SLAVERY → COLONIAL which was interesting because it refered not to Colonial Club, the eating club which we added to the Princeton words, but rather to the other definition of COLONIAL. Codenames is most often fun because ambiguity and flexibility, with lots of their words having multiple meanings to be able to be used in different contexts. Princeton words do have less of this flexibility.

Some less favorable clues mentioned were ones which in none of the remaining words seemed to make sense, but it was a bit difficult to describe them, because there was not an easy way to convey to the user what the clues should have been related to at the end of the game. One user noted that they had a clue CUI, which they did not know what that meant. This perhaps suggests that there can still be improvement made on how we are filtering the words based on common knowledge and frequency.

Users also noted that the model gives clues that relate to 1 word very often. Based on our implementation, this means that there were no word pairs with a similarity threshold over 0.25, meaning that the words on the board are quite diverse. One more aspect of the game that gave users some difficulty is that if the user is not able the guess the word(s) for the clue the model provides, and the board remains largely the same after the opponent takes a turn, our model will output the same clue again the next round

(because our clue generation is deterministic). We know this is a limitation of our game, but could not devise an easy way to adapt or modify the clue, given that user may not understand it, unlike the way a human might be able to adapt.

Overall, users found the game quite fun and found that the clues were generally good, but not "necessarily in a way a human might think." This definitely an important point worth mentioning: even though our model uses word embeddings trained on a broad language context, it lacks more interpersonal context as well as pop culture and inside jokes or references that human players might exploit to play this game more effectively. Those are some of the reasons why Codenames is fun to play. Players also make 2–3 card clues more often, in attempts to link the words in a creative or personal way that they hope the other player understands.

6 Conclusion & Future Work

Our project ultimately demonstrates several key findings. First, it *is* possible to adapt Codenames for a specific branch of language. In this case, we demonstrated that combining 20 Codenames board words with 5 Princeton-specific words led to a customized and engaging game. In future work, we could expand this finding to other varieties of language. For example, speakers of a regional or cultural dialects could add words specific to their dialect. Furthermore, this could help learners practice new vocabulary. Pre-med students, for instance, could play a version of Codenames full of medical jargon as a fun way to study for the MCAT. The typical word-embedding-based clue generation technique resulted in fairly successful clues, but other, more complex methods, such as WordNet path traversal and multi-round strategies, are often even more successful. Future work could determine whether these techniques are also beneficial for customized versions of Codenames, particularly when certain specialized vocabulary might not appear in WordNet at all.

Second, we demonstrate that fine-tuning word embeddings can be an efficient and successful way to adapt existing language knowledge to a specialized dialect or field. In

this case, a corpus of 629,845 words proved sufficient, though a larger and more diverse set of Princeton training data would likely have improved the model. Again, this method could be applied to other examples of jargon and dialects, though a dialect, which is less similar to standard language use than just vocabulary/jargon variation, would likely require a significantly larger fine-tuning corpus.

Third, we identified an interesting pattern, in which clues for an entirely Princeton or entirely standard set of target words tended to be more successful (from the human user perspective) than clues attempting to relate to both Princeton and non-Princeton target words. This suggests a potential flaw in the fine-tuning word embeddings method, in which original words potentially appear in an entirely different area of the word embedding vector space than new or updated words. This phenomenon, and potential solutions, certainly warrant further study.

Overall, our custom Codenames clue generator was successful. Quantitatively, it resulted in clues far more similar to target words than to other words on the board, and produced clues of reasonable sizes. Qualitatively, users reported that the clues were good but occasionally confusing. They lacked the kind of human perspective that may make these word games fun. This makes sense and suggests that perhaps current machine translation technologies may still be far from human-like understanding when it comes to word association games like Codenames that may require a close interpersonal connections.

7 Honor Code

This paper represents our own work in accordance with University regulations.

/s/ Julia Ying

/s/ Nora Graves

8 Contributions

Julia Ying Scraped articles from Wikipedia; curated list of Princeton Codenames words; manually found articles from *The Daily Princetonian* to scrape; calculated Princeton word frequencies; implemented GloVe Wiki Gigaword 100 and 300; debugged and finished training word embeddings model; set up GitHub repository; built Codenames full-stack website; linked front-end to existing back-end for website; deployed website; prepared slides for presentation; created user survey then conducted user evaluations on clue quality; wrote paper's Approach, Codenames rules in Introduction & Motivation, Princeton-related Data Curation, Codenames Simulator Implementation, User Studies, Appendices A.3–A.5; touched up and submitted paper.

Nora Graves Created framework for scraping articles from *The Daily Princetonian* using `BeautifulSoup`; cleaned Princeton-related corpus; set up Word2Vec and word embeddings model; recalculated frequencies of each word in vocabulary; wrote clue generation and opponent game play code, which later became the back-end for website; tested similarity threshold hyperparameter; created slides for presentation; prepared slides for presentation; conducted quantitative analysis on clue quality; wrote paper's Introduction & Motivation, Related Work, Word Embeddings Models Results, Clue Scoring & Ranking, Quantitative Evaluation, Conclusion & Future Work, Appendices A.1–A.2.

References

[1] Christopher Archibald and Diego Blaylock. "Noisy Communication Modeling for Improved Cooperation in Codenames". In: *2024 IEEE Conference on Games (CoG)*. 2024 IEEE Conference on Games (CoG). ISSN: 2325-4289. Aug. 2024, pp. 1–8. DOI: 10.1109/CoG60054.2024.10645589. URL: https://ieeexplore.ieee.org/abstract/document/10645589 (visited on 04/21/2025).

[2] Zhenisbek Assylbekov and Rustem Takhanov. *Context Vectors are Reflections of Word Vectors in Half the Dimensions*. 2019. arXiv: 1902.09859 [stat.ML]. URL: https://arxiv.org/abs/1902.09859.

[3] Joseph Bills and Christopher Archibald. "A Deductive Agent Hierarchy: Strategic Reasoning in Codenames". In: *2023 IEEE Conference on Games (CoG)*. 2023 IEEE Conference on Games (CoG). ISSN: 2325-4289. Aug. 2023, pp. 1–8. DOI: 10.1109/CoG57401.2023.10333226. URL: https://ieeexplore.ieee.org/abstract/document/10333226 (visited on 04/21/2025).

[4] Piotr Bojanowski et al. *Enriching Word Vectors with Subword Information*. 2017. arXiv: 1607.04606 [cs.CL]. URL: https://arxiv.org/abs/1607.04606.

[5] Réka Cserháti et al. "Codenames as a Game of Co-occurrence Counting". In: *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*. CMCL 2022. Ed. by Emmanuele Chersoni et al. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 43–53. DOI: 10.18653/v1/2022.cmcl-1.5. URL: https://aclanthology.org/2022.cmcl-1.5/ (visited on 04/21/2025).

[6] Daily Princetonian. *The Frosh Dictionary*. URL: https://frosh.dailyprincetonian.com/frosh-dictionary.html (visited on 04/15/2025).

[7] DataStax. *What is Cosine Similarity: A Comprehensive Guide*. URL: https://www.datastax.com/guides/what-is-cosine-similarity.

[8] Micha De Rijk and David Mareček. "Using Word Embeddings and Collocations for Modelling Word Associations". In: *Prague Bulletin of Mathematical Linguistics* 114.1 (Apr. 2020), pp. 35–57. ISSN: 1804-0462, 0032-6585. DOI: 10.14712/00326585.002. URL: http://ufal.mff.cuni.cz/pbml/114/art-de-rijk-marecek (visited on 04/21/2025).

[9] Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. "Eigenwords: Spectral Word Embeddings". In: *Journal of Machine Learning Research* 16.95 (2015), pp. 3035–3078. URL: http://jmlr.org/papers/v16/dhillon15a.html.

[10] E. H. Babbit. "College Words and Phrases". In: *Dialect Notes*. Vol. II. Google-Books-ID: vbRZAAAAMAAJ. American Dialect Society, 1904. URL: https://books.google.com/books?id=vbRZAAAAMAAJ&pg=PA3&source=gbs_toc_r&cad=2#v=onepage&q&f=false.

[11] Czech Games Edition. *Codenames*. URL: https://www.czechgames.com/games/codenames.

[12] Eric Eisenberg. *Crunching the Numbers: The Stats Behind Popular Word Games*. Medium. Aug. 3, 2023. URL: https://wordgames.medium.com/crunching-the-numbers-the-stats-behind-popular-word-games-d0827a3af601 (visited on 04/15/2025).

[13] Tahereh Firoozi et al. "The Effect of Fine-tuned Word Embedding Techniques on the Accuracy of Automated Essay Scoring Systems Using Neural Networks". In: *Journal of Applied Testing Technology* (Dec. 16, 2022), pp. 21–29. URL: http://jattjournal.net/index.php/atp/article/view/172687 (visited on 04/15/2025).

[14] Matthew Freestone and Shubhra Kanti Karmaker Santu. *Word Embeddings Revisited: Do LLMs Offer Something New?* 2024. arXiv: 2402.11094 [cs.CL]. URL: https://arxiv.org/abs/2402.11094.

[15] Kenneth R. Ginsburg, and the Committee on Communications, and and the Committee on Psychosocial Aspects of Child and Family Health. "The Importance of Play in Promoting Healthy Child Development and Maintaining Strong Parent-Child Bonds". In: *Pediatrics* 119.1 (Jan. 1, 2007), pp. 182–191. ISSN: 0031-4005, 1098-4275. DOI: 10.1542/peds.2006-2697. URL: https://publications.aap.org/pediatrics/article/119/1/182/70699/The-Importance-of-Play-in-Promoting-Healthy-Child (visited on 04/15/2025).

[16] Ruth Graham. "The Perfect Balance Between Lonely Agony and Raucous Collaboration". In: *Slate* (Nov. 28, 2018). Section: Picks. ISSN: 1091-2339. URL: https://slate.com/human-interest/2018/11/best-family-games-codenames-rules.html (visited on 04/15/2025).

[17] Hamza Jamal Jassim. "Language and Identity: How Dialects Shape Social Perceptions". In: (Dec. 1, 2024), pp. 1–23.

[18] Catalina Jaramillo et al. "Word Autobots: Using Transformers for Word Association in the Game Codenames". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16.1 (Oct. 1, 2020). Number: 1, pp. 231–237. ISSN: 2334-0924. DOI: 10.1609/aiide.v16i1.7435. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/7435 (visited on 04/21/2025).

[19] Niklas Johannes, Matti Vuorre, and Andrew K. Przybylski. *Video game play is positively correlated with well-being*. Nov. 13, 2020. DOI: 10.31234/osf.io/qrjza. URL: https://osf.io/qrjza_v1 (visited on 04/15/2025).

[20] Irving J. Spitzberg Jr and Virginia V. Thorndike. *Creating Community on College Campuses*. Google-Books-ID: J2mvLUoQ46MC. State University of New York Press, July 1, 1992. 264 pp. ISBN: 978-1-4384-2081-3.

[21] Andrew Kim et al. "Cooperation and Codenames: Understanding Natural Language Processing via Codenames". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 15.1 (Oct. 8, 2019). Number: 1, pp. 160–166. ISSN: 2334-0924. DOI: 10.1609/aiide.v15i1.5239. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/5239 (visited on 04/21/2025).

[22] Kanako Komiya and Hiroyuki Shinnou. "Investigating Effective Parameters for Fine-tuning of Word Embeddings Using Only a Small Corpus". In: *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*. ACL 2018. Ed. by Reza Haffari et al. Melbourne: Association for Computational Linguistics, July 2018, pp. 60–67. DOI: 10.18653/v1/W18-3408. URL: https://aclanthology.org/W18-3408/ (visited on 04/15/2025).

[23] Divya Koyyalagunta et al. "Playing Codenames with Language Graphs and Word Embeddings". In: *Journal of Artificial Intelligence Research* 71 (June 23, 2021), pp. 319–346. ISSN: 1076-9757. DOI: 10.1613/jair.1.12665. URL: https://jair.org/index.php/jair/article/view/12665 (visited on 04/21/2025).

[24] Kong Wei Kun et al. "KGWE: A Knowledge-guided Word Embedding Fine-tuning Model". In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). ISSN: 2375-0197. Nov. 2021, pp. 1221–1225. DOI: 10.1109/ICTAI52525.2021.00193. URL: https://ieeexplore.ieee.org/document/9643233/ (visited on 04/15/2025).

[25] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[26] Madison R. Myers-Burg and Douglas A. Behrend. "More than just accent? The role of dialect words in children's language-based social judgments". In: *Journal of Experimental Child Psychology* 204 (Apr. 1, 2021), p. 105055. ISSN: 0022-0965. DOI: 10.1016/j.jecp.2020.105055. URL: https://www.sciencedirect.com/science/article/pii/S0022096520305099 (visited on 04/15/2025).

[27] Korawit Orkphol and Wu Yang. "Word Sense Disambiguation Using Cosine Similarity Collaborates with Word2vec and WordNet". In: *Future Internet* 5 (2019). ISSN: 1999-5903. DOI: 10.3390/fi11050114. URL: https://www.mdpi.com/1999-5903/11/5/114.

[28] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[29] Alina Petukhova, João P. Matos-Carvalho, and Nuno Fachada. *Text Clustering with LLM Embeddings*. 2024. arXiv: 2403.15112 [cs.CL]. URL: https://arxiv.org/abs/2403.15112.

[30] The Daily Princetonian. *About Us*. URL: https://www.dailyprincetonian.com/page/about.

[31] The Daily Princetonian. *The First Year Dictionary*. URL: https://frosh.dailyprincetonian.com/frosh-dictionary.html.

[32] Rachael Tatman. *English Word Frequency*. Kaggle. URL: https://www.kaggle.com/datasets/rtatman/english-word-frequency (visited on 04/21/2025).

[33] Giorgia Ramponi et al. "Vocabulary-based community detection and characterization". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. New York, NY, USA: Association for Computing Machinery, Apr. 8, 2019, pp. 1043–1050. ISBN: 978-1-4503-5933-7. DOI: 10.1145/3297280.3297384. URL: https://dl.acm.org/doi/10.1145/3297280.3297384 (visited on 04/15/2025).

[34] Matthew Stephenson, Matthew Sidji, and Benoît Ronval. *Codenames as a Benchmark for Large Language Models*. Dec. 16, 2024. DOI: 10.48550/arXiv.2412.11373. arXiv: 2412.11373[cs]. URL: http://arxiv.org/abs/2412.11373 (visited on 04/15/2025).

[35] Kejkaew Thanasuan and Shane T. Mueller. "Improving Lexical Memory Access and Decision Making Processes Using Cognitive Word Games". In: *Proceedings of the Annual Meeting of the Cognitive Science Society* 37.0 (2015). URL: https://escholarship.org/uc/item/6pj47967 (visited on 04/15/2025).

[36] Tan Thongtan and Tanasanee Phienthrakul. "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Ed. by Fernando Alva-Manchego, Eunsol Choi, and Daniel Khashabi. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 407–414. DOI: 10.18653/v1/P19-2057. URL: https://aclanthology.org/P19-2057.

[37] Julien Tissier, Christophe Gravier, and Amaury Habrard. "Dict2vec : Learning Word Embeddings using Lexical Dictionaries". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2017. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Asso-

ciation for Computational Linguistics, Sept. 2017, pp. 254–263. DOI: 10.18653/v1/D17-1024. URL: https://aclanthology.org/D17-1024/ (visited on 04/20/2025).

[38] Xiao-Kun Wu et al. "LLM Fine-Tuning: Concepts, Opportunities, and Challenges". In: *Big Data and Cognitive Computing* 9.4 (Apr. 2025). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 87. ISSN: 2504-2289. DOI: 10.3390/bdcc9040087. URL: https://www.mdpi.com/2504-2289/9/4/87 (visited on 04/15/2025).

[39] Xuewei Yan, Yongqing Li, and Cameron Shaw. "Codenames_AI_Report". In: ().

[40] Xuefeng Yang and Kezhi Mao. "Task Independent Fine Tuning for Word Embeddings". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.4 (Apr. 2017), pp. 885–894. ISSN: 2329-9304. DOI: 10.1109/TASLP.2016.2644863. URL: https://ieeexplore.ieee.org/abstract/document/7797241 (visited on 04/15/2025).

[41] Michael Yogman et al. "The Power of Play: A Pediatric Role in Enhancing Development in Young Children". In: *Pediatrics* 142.3 (Sept. 1, 2018), e20182058. ISSN: 0031-4005. DOI: 10.1542/peds.2018-2058. URL: https://doi.org/10.1542/peds.2018-2058 (visited on 04/15/2025).

[42] Zixiao Zhu and Kezhi Mao. "Knowledge-based BERT word embedding fine-tuning for emotion recognition". In: *Neurocomputing* 552 (Oct. 1, 2023), p. 126488. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2023.126488. URL: https://www.sciencedirect.com/science/article/pii/S0925231223006112 (visited on 04/15/2025).

A Appendix

A.1 Original Codenames Words

AFRICA, AGENT, AIR, ALIEN, ALPS, AMAZON, AMBULANCE, AMERICA, ANGEL, ANTARCTICA, APPLE, ARM, ATLANTIS, AUSTRALIA, AZTEC, BACK, BALL, BAND, BANK, BAR, BARK, BAT, BATTERY, BEACH, BEAR, BEAT, BED, BEIJING, BELL, BELT, BERLIN, BERMUDA, BERRY, BILL, BLOCK, BOARD, BOLT, BOMB, BOND, BOOM, BOOT, BOTTLE, BOW, BOX, BRIDGE, BRUSH, BUCK, BUFFALO, BUG, BUGLE, BUTTON, CALF, CANADA, CAP, CAPITAL, CAR, CARD, CARROT, CASINO, CAST, CAT, CELL, CENTAUR, CENTER, CHAIR, CHANGE, CHARGE, CHECK, CHEST, CHICK, CHINA, CHOCOLATE, CHURCH, CIRCLE, CLIFF, CLOAK, CLUB, CODE, COLD, COMIC, COMPOUND, CONCERT, CONDUCTOR, CONTRACT, COOK, COPPER, COTTON, COURT, COVER, CRANE, CRASH, CRICKET, CROSS, CROWN, CYCLE, CZECH, DANCE, DATE, DAY, DEATH, DECK, DEGREE, DIAMOND, DICE, DINOSAUR, DISEASE, DOCTOR, DOG, DRAFT, DRAGON, DRESS, DRILL, DROP, DUCK, DWARF, EAGLE, EGYPT, EMBASSY, ENGINE, ENGLAND, EUROPE, EYE, FACE, FAIR, FALL, FAN, FENCE, FIELD, FIGHTER, FIGURE, FILE, FILM, FIRE, FISH, FLUTE, FLY, FOOT, FORCE, FOREST, FORK, FRANCE, GAME, GAS, GENIUS, GERMANY, GHOST, GIANT, GLASS, GLOVE, GOLD, GRACE, GRASS, GREECE, GREEN, GROUND, HAM, HAND, HAWK, HEAD, HEART, HELICOPTER, HIMALAYAS, HOLE, HOLLYWOOD, HONEY, HOOD, HOOK, HORN, HORSE, HORSESHOE, HOSPITAL, HOTEL, ICE, ICE CREAM,[7] INDIA, IRON, IVORY, JACK, JAM, JET, JUPITER, KANGAROO, KETCHUP, KEY, KID, KING, KIWI, KNIFE, KNIGHT, LAB, LAP, LASER, LAWYER, LEAD, LEMON, LEPRECHAUN, LIFE, LIGHT, LIMOUSINE, LINE, LINK, LION, LITTER, LOCH NESS,[7] LOCK, LOG, LONDON, LUCK, MAIL, MAMMOTH, MAPLE, MARBLE, MARCH, MASS, MATCH, MERCURY, MEXICO, MICROSCOPE, MILLIONAIRE, MINE, MINT, MISSILE, MODEL, MOLE, MOON, MOSCOW, MOUNT, MOUSE,

[7]Removed from final game (does not appear in pre-trained word embeddings).

MOUTH, MUG, NAIL, NEEDLE, NET, NEW YORK, [7] NIGHT, NINJA, NOTE, NOVEL, NURSE, NUT, OCTOPUS, OIL, OLIVE, OLYMPUS, OPERA, ORANGE, ORGAN, PALM, PAN, PANTS, PAPER, PARACHUTE, PARK, PART, PASS, PASTE, PENGUIN, PHOENIX, PIANO, PIE, PILOT, PIN, PIPE, PIRATE, PISTOL, PIT, PITCH, PLANE, PLASTIC, PLATE, PLATYPUS, PLAY, PLOT, POINT, POISON, POLE, POLICE, POOL, PORT, POST, POUND, PRESS, PRINCESS, PUMPKIN, PUPIL, PYRAMID, QUEEN, RABBIT, RACKET, RAY, REVOLUTION, RING, ROBIN, ROBOT, ROCK, ROME, ROOT, ROSE, ROULETTE, ROUND, ROW, RULER, SATELLITE, SATURN, SCALE, SCHOOL, SCIENTIST, SCORPION, SCREEN, SCUBA DIVER,[7] SEAL, SERVER, SHADOW, SHAKESPEARE, SHARK, SHIP, SHOE, SHOP, SHOT, SINK, SKYSCRAPER, SLIP, SLUG, SMUGGLER, SNOW, SNOWMAN, SOCK, SOLDIER, SOUL, SOUND, SPACE, SPELL, SPIDER, SPIKE, SPINE, SPOT, SPRING, SPY, SQUARE, STADIUM, STAFF, STAR, STATE, STICK, STOCK, STRAW, STREAM, STRIKE, STRING, SUB, SUIT, SUPERHERO, SWING, SWITCH, TABLE, TABLET, TAG, TAIL, TAP, TEACHER, TELESCOPE, TEMPLE, THEATER, THIEF, THUMB, TICK, TIE, TIME, TOKYO, TOOTH, TORCH, TOWER, TRACK, TRAIN, TRIANGLE, TRIP, TRUNK, TUBE, TURKEY, UNDERTAKER, UNICORN, VACUUM, VAN, VET, WAKE, WALL, WAR, WASHER, WASHINGTON, WATCH, WATER, WAVE, WEB, WELL, WHALE, WHIP, WIND, WITCH, WORM, YARD

A.2 Princeton Codenames Words

NASSAU, FIRESTONE, FRIST, LATE MEAL, REUNIONS, ORANGE BUBBLE, HONOR CODE, RCA, RESIDENTIAL COLLEGE, PSAFE, LAKE CARNEGIE, ZEE, PDF, PROX, OA, EATING CLUB, BICKER, IVY, COTTAGE, CANNON, CHARTER, COLONIAL, TERRACE, QUAD, CLOISTER, TIGER INN, CAP AND GOWN, TOWER, CAMPUS CLUB, FIZZ, TIGERAPPS, PROSPECT, LAWNPARTIES, USG, MCCOSH, TIGERTRANSIT, OLD NASSAU, U-STORE, WAWA, EAST PYNE, FRICK, LEWIS, CANNON GREEN, LCA, ROBERTSON, COS, SPIA, MCCARTER, JADWIN, DILLON, FORBES, WHITMAN, BUTLER, ROCKY, MATHEY, YEH, NCW, CJL, EQUAD,

FRIEND CENTER, DINKY, PRINCETON JUNCTION, ROOM DRAW, POE FIELD, LABYRINTH, HOAGIE, JUNBI, BENT SPOON, SMALL WORLD, FINE HALL, COFFEE CLUB, NJ TRANSIT, TRIANGLE CLUB, EISGRUBER, PRE-RADE, DAILY PRINCETONIAN

A.3 Princeton Codenames Similar Words Based on Model

Word	**Top 10 Similar Words (with similarity score)**
nassau	east_pyne (0.518), murray-dodge (0.454), located (0.449), downtown (0.437), county (0.437), brooklyn (0.409), hall (0.407), beach (0.403), 5{6 (0.403), florida (0.395)
firestone	tire (0.545), ford (0.408), recalled (0.328), library (0.326), bent_spoon (0.326), friend_center (0.317), invitational (0.317), east_pyne (0.315), lights (0.309), woods (0.298)
frist	r (0.419), late_meal (0.416), house (0.387), center (0.385), senate (0.367), senator (0.365), mccosh (0.358), heart (0.355), republican (0.348), health (0.346)
late_meal	grab-and-go (0.645), meal (0.579), meals (0.578), dining (0.567), options (0.554), breakfast (0.515), plan. (0.508), roastery (0.500), shopping (0.496), c-store (0.491)
reunions	dinners (0.390), concerts (0.375), meetings (0.362), korean (0.353), attend (0.349), conferences (0.349), families (0.348), seminars (0.331), classes (0.329), informal (0.325)

orange_bubble	numine (0.550), newbert (0.533), acceptances. (0.533), b.s.e (0.532), viget (0.529), triangle_club (0.526), fine_hall (0.526), mind (0.524), davis_20.23 (0.521), serveries (0.516)
honor_code	responsible (0.511), violations (0.491), violation (0.489), instances (0.478), 51 (0.473), cases (0.462), 2019{2024 (0.450), committee (0.439), regarding (0.435), reported (0.423)
rca	recording (0.500), studio (0.447), records (0.411), recorded (0.405), debut (0.390), 1969 (0.365), 1964 (0.365), singles (0.359), singer (0.356), 1965 (0.355)
residential_college	undergraduate (0.677), disciplinary (0.669), board (0.598), dean (0.593), faculty (0.506), waitlist (0.474), graduate (0.462), discipline (0.462), upperclass (0.412), colleges (0.409)
psafe	officers (0.663), pulled (0.535), walked (0.535), protesters (0.525), fine/mcdonnell (0.522), gathered (0.510), police (0.505), pre-pandemic (0.502), demonstrators (0.500), 10. (0.478)
lake_carnegie	poe_field (0.687), 2008-10-03 (0.652), down-campus (0.542), puam (0.541), cannon_green (0.539), serveries (0.526), equad (0.519), 23. (0.516), hugas (0.509), ecampus (0.507)
zee	one-on-ones (0.351), tv (0.345), bin (0.343), oh (0.325), - (0.300), siddiqui (0.287), yeh (0.279), al (0.273), am (0.270)

pdf	file (0.524), document (0.480), format (0.467), copy (0.452), archived (0.411), application (0.410), text (0.408), enables (0.396), users (0.395), web (0.387)
prox	id (0.259), cards (0.257), card (0.251), enables (0.226), postdocs (0.223), defeatism (0.217), 5. (0.206), roulette (0.203), grippo (0.203), virtual (0.202)
oa	lodge (0.329), rec (0.326), hugas (0.309), triangle_club (0.289), davis_20.23 (0.285), kappa (0.276), phi (0.276), cpuc (0.276), odus (0.275), internship (0.273)
eating_club	club (0.476), eating_clubs (0.455), membership (0.447), option (0.445), choice (0.438), tiger_inn (0.429), clubs (0.424), join (0.417), individual (0.409), interest (0.397)
bicker	bickered (0.486), sign-in (0.474), tiger_inn (0.442), bickering (0.414), clubs (0.397), pre-draw (0.384), parties (0.375), bickerees (0.369), selective (0.368), process (0.360)
ivy	league (0.438), collegiate (0.388), conference (0.382), schools (0.380), ivies (0.373), intercollegiate (0.372), colleges (0.372), tournament (0.359), undefeated (0.358), ncaa (0.358)
cottage	inn (0.605), terrace (0.548), cap_and_gown (0.543), colonial (0.474), lodge (0.451), residence (0.448), house (0.445), tiger_inn (0.434), brick (0.430), houses (0.428)

cannon	fired (0.453), cap_and_gown (0.445), tiger_inn (0.417), machine (0.410), tower (0.402), hotchkiss (0.379), club (0.372), pistol (0.361), cottage (0.358), 2001. (0.355)
charter	tiger_inn (0.501), membership (0.469), accepted (0.406), cap_and_gown (0.395), elected (0.394), club (0.370), founding (0.363), colonial (0.362), adopted (0.356), signed (0.351)
colonial	colonies (0.480), cottage (0.474), terrace (0.459), 19th (0.459), cap_and_gown (0.450), tiger_inn (0.450), era (0.441), tower (0.439), british (0.434), inn (0.431)
terrace	cottage (0.548), tower (0.526), cap_and_gown (0.517), courtyard (0.506), inn (0.486), plaza (0.481), lounge (0.477), avenue (0.462), colonial (0.459), hotel (0.458)
quad	quads (0.582), triple (0.450), terrace (0.445), 3{0 (0.406), loop (0.405), double (0.405), tower (0.396), foot (0.381), jump (0.363), equad (0.362)
cloister	tower (0.477), chapel (0.439), terrace (0.437), courtyard (0.419), inn (0.418), facade (0.408), renovated (0.394), gothic (0.391), church (0.387), dormitory (0.383)
tiger_inn	cap_and_gown (0.683), bickerees (0.607), accepted (0.607), club (0.604), 2001. (0.590), clubs (0.515), tower (0.505), charter (0.501), bickeree (0.482), membership (0.474)

cap_and_gown	tiger_inn (0.683), gown (0.595), club (0.587), cottage (0.543), tower (0.519), terrace (0.517), bickerees (0.510), 2001. (0.486), accepted (0.477), bickeree (0.457)
tower	terrace (0.526), skyscraper (0.524), cap_and_gown (0.519), tiger_inn (0.505), building (0.483), cloister (0.477), stands (0.467), hotel (0.464), plaza (0.448), bickerees (0.445)
campus_club	down-campus (0.553), worldlings (0.549), c-store (0.548), dailyprincetonian.com (0.541), hugas (0.541), kickline (0.537), wristbanding (0.533), serveries (0.528), choppa (0.509), 2008-10-03 (0.502)
fizz	bottle (0.316), ketchup (0.314), caffeine (0.300), sip (0.290), lil (0.288), pop (0.275), iced (0.274), drink (0.274), flavor (0.267), syrup (0.259)
tigerapps	tigerjunction (0.722), yengibaryan (0.616), c-store (0.615), freefood (0.609), b.s.e (0.600), kickline (0.592), 2008-10-03 (0.585), laufey (0.584), geo-exchange (0.582), serveries (0.580)
prospect	news (0.453), avenue (0.430), hartnett-cody (0.426), writer (0.424), editor (0.421), contributing (0.419), possibility (0.413), facing (0.389), future (0.388)
lawnparties	spring (0.591), fall (0.570), beginning (0.522), semester (0.521), 2019. (0.511), 2021. (0.502), autumn (0.501), 2022. (0.492), last (0.491), 2024. (0.489)

usg	government (0.489), eisgruber (0.401), approved (0.372), meeting (0.372), council (0.369), 2013. (0.364), chair (0.363), u-council (0.353), unanimously (0.346), swamidurai (0.342)
mccosh	east_pyne (0.578), courtyard (0.565), 10. (0.521), hall (0.475), equad (0.463), halls (0.458), 5{6 (0.443), protesters (0.431), 016. (0.422), auditorium (0.422)
tigertransit	routes (0.667), route (0.655), buses (0.618), operate (0.595), nj_transit (0.568), transit (0.567), bus (0.554), service (0.547), rail (0.544), operates (0.541)
old_nassau	worldling (0.517), siyeonlee (0.504), serveries (0.495), cdrobles (0.488), long (0.487), kickline (0.458), downcampus (0.453), lives (0.447), b.s.e (0.447), numine (0.443)
u-store	c-store (0.670), princeton_junction (0.590), store (0.522), bent_spoon (0.516), roastery (0.512), grocery (0.500), worldlings (0.499), located (0.493), coffee_club (0.490), stores (0.484)
wawa	convenience (0.387), grocery (0.328), shopping (0.323), stores (0.320), store (0.309), stations (0.299), starbucks (0.299), shopper (0.289), station (0.289), u-store (0.286)
east_pyne	murray-dodge (0.686), down-campus (0.594), mccosh (0.578), fine_hall (0.569), courtyard (0.541), 016. (0.541), equad (0.532), hall (0.524), 10. (0.522), 5{6 (0.521)

frick	museum (0.327), carnegie (0.317), c. (0.301), ford (0.300), henry (0.287), garrett (0.283), gallery (0.282), collection (0.280), e. (0.270), lillian (0.269)
lewis	johnson (0.551), jr. (0.540), jackson (0.538), taylor (0.532), hamilton (0.530), stewart (0.518), james (0.513), smith (0.503), williams (0.502), anderson (0.501)
cannon_green	2008-10-03 (0.594), 5{6 (0.561), lake_carnegie (0.539), 23. (0.535), monday (0.520), poe_field (0.514), wintersession (0.509), sunday (0.502), saturday (0.489), friday (0.485)
lca	vision (0.268), boardings (0.264), program (0.262), project (0.246), cps (0.239), st. (0.231), cycles (0.231), st (0.218), vitus (0.216), craft (0.213)
robertson	stephens (0.630), walker (0.548), alexander (0.499), 016. (0.491), james (0.487), campbell (0.484), smith (0.470), graham (0.451), murphy (0.445), walter (0.441)
cos	inc (0.329), companies (0.292), amp (0.290), na (0.288), categories (0.287), specialty (0.284), cp (0.274), x (0.271), u (0.264), differently (0.264)
spia	pass/d/fail (0.493), davis_20.23 (0.491), odus (0.488), cjl (0.484), yengibaryan (0.475), baudez (0.474), siyeonlee (0.460), blumauer (0.458), techspark (0.457), affairs (0.454)
mccarter	theater (0.327), n.j. (0.311), theatre (0.304), mercer (0.287), princeton (0.286), rutgers (0.285), trenton (0.279), newark (0.278), firm (0.261), nj (0.254)

jadwin	gymnasium (0.450), gym (0.422), games (0.361), murray-dodge (0.352), east_pyne (0.323), down-campus (0.316), home (0.314), equad (0.304), shea (0.302), rutgers (0.298)
dillon	jacobs (0.430), stephens (0.364), chase (0.354), morgan (0.353), ryan (0.345), matt (0.337), edwards (0.330), smith (0.329), anderson (0.326), lynch (0.312)
forbes	whitman (0.556), equad (0.493), alexander (0.445), c-store (0.406), butler (0.405), mathey (0.395), hampshire (0.394), bradley (0.388), 016. (0.385), robertson (0.384)
whitman	forbes (0.556), equad (0.450), college (0.415), butler (0.415), jersey (0.407), pennsylvania (0.392), hampshire (0.391), dickinson (0.390), residential_colleges (0.385), house (0.380)
butler	anderson (0.485), richardson (0.464), walker (0.463), 016. (0.460), james (0.459), parker (0.448), pierce (0.446), xaivian (0.439), baker (0.439), francis (0.437)
rocky	cliff (0.469), meadows (0.440), adjacent (0.435), down-campus (0.427), hill (0.419), mount (0.408), steep (0.404), stretches (0.403), rock (0.398), mathey (0.388)
mathey	residential_colleges (0.424), college (0.421), forbes (0.395), butler (0.389), rocky (0.388), colleges (0.388), whitman (0.370), graduates (0.343), yeh (0.341), ncw (0.339)

yeh	chang (0.446), wang (0.436), chen (0.436), yi (0.424), yu (0.402), wu (0.391), yang (0.370), zhang (0.359), lee (0.355), ma (0.348)
ncw	whitman (0.348), college (0.346), residential_colleges (0.340), mathey (0.339), butler (0.328), hampshire (0.311), newest (0.298), forbes (0.296), puam (0.286), fine/mcdonnell (0.284)
cjl	steinlauf (0.561), davis_20.23 (0.560), yengibaryan (0.545), jewish (0.540), freefood (0.535), chabad (0.526), 2008-10-03 (0.524), c-store (0.519), kickline (0.512), siyeonlee (0.503)
equad	friend_center (0.666), princeton_junction (0.646), 5{6 (0.615), downcampus (0.596), c-store (0.592), walk (0.560), 4. (0.555), 2008-10-03 (0.537), down-campus (0.534), residential_colleges (0.532)
friend_center	equad (0.666), 4 (0.633), route (0.632), princeton_junction (0.620), fine/mcdonnell (0.599), 1 (0.587), station (0.577), 2 (0.565), routes (0.557), bus (0.543)
dinky	cute (0.350), tots (0.342), fancy (0.325), pastries (0.319), snacks (0.287), totes (0.264), homework (0.260), sandwiches (0.260), snack (0.258), fried (0.257)
princeton_junction	equad (0.646), friend_center (0.620), 5{6 (0.614), route (0.611), station (0.608), c-store (0.604), 4. (0.593), u-store (0.590), a.m. (0.568), 4 (0.565)

room_draw	pre-draw (0.663), begin (0.532), draw (0.529), starting (0.515), drawmate (0.472), 10. (0.468), semester (0.455), phase (0.454), session (0.447), beginning (0.446)
poe_field	lake_carnegie (0.687), site (0.624), construction (0.607), located (0.579), adjacent (0.561), 2008-10-03 (0.554), down-campus (0.544), area (0.531), nearby (0.529), puam (0.523)
labyrinth	underground (0.377), hidden (0.369), pan (0.348), theater (0.332), secret (0.326), convoluted (0.321), basement (0.321), courtyard (0.321), passages (0.309), book (0.294)
hoagie	sandwich (0.388), roll (0.335), cheese (0.326), sandwiches (0.312), bread (0.283), taco (0.281), pizza (0.277), bagels (0.269), sushi (0.250), dough (0.250)
junbi	cdrobles (0.532), tigerdraw (0.525), bent_spoon (0.523), kickline (0.501), downcampus (0.500), siyeonlee (0.497), delizioso (0.496), 3{0 (0.492), 4. (0.490), c-store (0.490)
bent_spoon	c-store (0.666), delizioso (0.630), serveries (0.605), b.s.e (0.594), worldlings (0.593), kickline (0.581), cdrobles (0.571), down-campus (0.567), siyeonlee (0.564), gheorghita (0.562)
small_world	shop (0.692), coffee (0.648), restaurant (0.547), cafe (0.532), coffee_club (0.520), store (0.512), roastery (0.506), shops (0.490), delizioso (0.476), bakery (0.468)

fine_hall	down-campus (0.796), hugas (0.739), serveries (0.739), yengibaryan (0.718), c-store (0.713), worldlings (0.709), b.s.e (0.708), schools{schools (0.699), siyeonlee (0.699), murray-dodge (0.694)
coffee_club	delizioso (0.653), roastery (0.568), maruichi (0.564), small_world (0.520), shop (0.510), restaurant (0.493), bent_spoon (0.490), u-store (0.490), coffee (0.486), kickline (0.481)
nj_transit	tigerjunction (0.605), roastery (0.587), tigertransit (0.568), coursebook (0.555), tigersnatch (0.538), serveries (0.509), p-rade (0.505), kickline (0.505), drawmate (0.502), b+ (0.498)
triangle_club	kickline (0.619), davis_20.23 (0.618), hugas (0.597), numine (0.594), viget (0.583), 2014. (0.583), down-campus (0.568), serveries (0.566), yengibaryan (0.566), 2008-10-03 (0.562)
eisgruber	president (0.662), christopher (0.614), vice (0.574), swamidurai (0.573), 2013. (0.544), bennett (0.524), baudez (0.518), spokesperson (0.518), statement (0.500), responded (0.498)
pre-rade	shortly (0.490), opening (0.489), 2023. (0.488), 2025. (0.477), 2026. (0.469), 5{6 (0.454), laufey (0.453), arrival (0.451), 2012. (0.450), february (0.448)
daily_princetonian	interview (0.559), { (0.543), interviewed (0.515), prince (0.503), told (0.496), statement (0.495), wrote (0.472), according (0.450), published (0.448)

A.4 Princeton Codenames Website

https://princetoncodenames.onrender.com/

A.5 Princeton Codenames GitHub Repository

https://github.com/juliaying26/PrincetonCodenames