(Spy)Mastering Codenames

By: Arjun Arora, Heidi Chen, Daniel Classon
CS 221 Fall 2018

Project Statement and Background

We will train a model to emulate the Spymaster in the game Codenames. In Codenames, two teams (red and blue) aim to guess the correct words from a grid. One person on each team, the Spymaster, knows the underlying locations of each team's words, and gives clues (i.e., ("ocean, 3") to help their team guess only their team's words, and not the others' or the "assassin" (instant-loss) square. Whichever team finds all their words first, wins.

Inputs, Outputs, and Scope

Our model's inputs will be a list of the blue words and lists of the undesired words--assassin words, the other team's words, and neutral words in descending order of cost. The output will be a single word, the 'hint', and the number of the blue words associated with the hint. Our model will maximize this number and the associations of blue words to the clue, while minimizing associations with words in the undesired list. We will further ensure that our model is better than the baseline (see below) by fine tuning a minimum threshold of association for the words. For example, a robot might be able to link 6 words based on obscure word features, but a human would only be able to link 3 using more prominent associations. For this project, we will set our scope to having an AI spymaster at least as good as a human spymaster; we will begin by building a spymaster that can calculate the optimal output for a single clue round, and determine from there if this heuristic reflects well in full games or if we need to incorporate more features in our decision making.

Evaluation

To evaluate this model, we will pair the Spymaster with a human team, and have them play against a human Spymaster and human team on a representative number of games. We will analyze metrics including the accuracy of human guesses based on bot clues, average clue sizes, win ratio for the bot versus human teams, etc.

Baseline & Oracle

For our baseline, we used a basic script written by Jeremy Neiman that uses the gensim package, a model which generates similar words based on a cluster of words with word2vec [1]. The baseline model does not optimize the hint based on a desired number of matches like our output will (i.e. [hint, # matches]) and only outputs a hint that attempts to best link *all* of the blue words (and often fails to link any blue words, as you will see). For our oracle, we determined clues manually based on our experience playing Codenames. We ran both the baseline and the oracle on the first turn of 10 Codenames boards and recorded the hints given as well as the board distribution, so that we can later run our model on the same example boards. A successful and failed example can be seen below:

Figure 1: Success.

For this round, our clue was “infrastructure, 2”, to match “fence” and “bridge.” The baseline’s clue was “public_works,” which we found very similar.

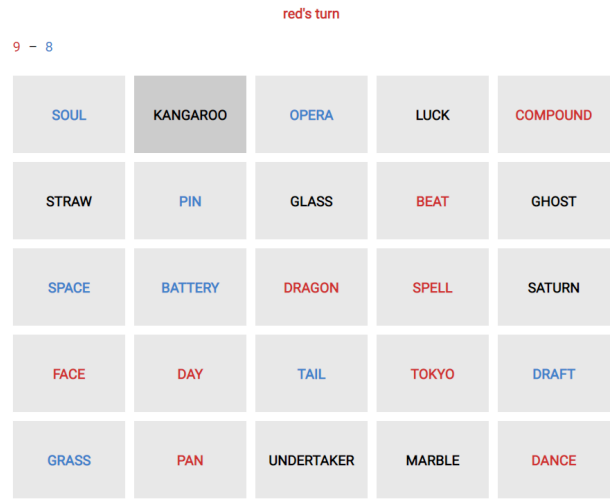

Figure 2: Failure.

Our clue was “Astroturf, 2” to indicate “space” and “grass”. The baseline’s clue was “easements”, which we had trouble associating with any of the blue words.

We rated each of the rounds “good”, “bad”, or “average”. Out of the 10 rounds, 3 were good, 2 were average, and 5 were bad (compared to what a human Codenames player would expect). We would rate our oracle 10/10.

Challenges

One of the challenges that we are confronted with is rating hints. Codenames is inherently subjective and the metrics that we tune our model to might not output “good hints” for anyone else. Additionally, one complex element of gameplay is deciding how many words a clue should be associated with. We will have to determine what the tradeoff is between maximizing the number of words encompassed by a clue, and the degree of association between each of those words and the clue. If we are able to accomplish our basic Spymaster model, there are considerations to be made in optimizing for winning full games, not just single rounds.

Possible Topics

Some possible topics that we might utilize for this project include loss minimization, stochastic gradient descent, linear or logistic regressions, and neural networks.

[1]: https://github.com/docmarionum1/codenames_blog/blob/master/Codenames%201%20-%20Word2Vec.ipynb