Simulating Codenames with Natural Language Processing
Machine Learning for Natural Language Processing 2021

Guillaume Hofmann
ENSAE, 3rd year
guillaume.hofmann@ensae.fr

Amandine Tran
ENSAE, 3rd year
amandine.tran@ensae.fr

Abstract

The main concern of this project [1] is to build NLP-based bots for the game *Codenames*, which requires a good understanding of words and their various meanings to win. Bots based on WordNet's synsets and various kinds of word embeddings are put to the test. Performances are assessed using a benchmark evaluating each team's win rates and overall greediness. In the end, it seems that GloVe-based algorithms show the best results. WordNet-based bots, relying exclusively on Wu-Palmer similarity, happen to also reach good performances by playing safer.

1 Problem Framing

Codenames is a popular board game which is based on similarity between words. Two teams compete, each team having one "codemaster" and one or several guessers. 25 words are placed on the table. The role of the guessers is to guess their team's words, while avoiding the words belonging to the other team, to the assassin or to civilian squares. The guessers guess their words based on single-word clues given by the codemaster. Therefore, this project aims at creating a bot which can play this game, either as a guesser or as the codemaster. More details about *Codenames* are available here.

2 Experiments Protocol

Different natural language processing approaches are used here :

- Synsets with WordNet,
- Word2Vec using the 300-dimension embedding pre-trained on the Google News corpus,
- GloVe embedding.

[1] Our project is available on this github.

The 25 word tiles are drawn from a given pool of 395 words. Bots which are based on Word2Vec or GloVe embedding follow similar logical processes, unlike those based on synsets:

- Guesser based on a pre-trained embedding uses the embedding's vocabulary to find the word(s) minimizing cosine distance to the clue word. Guesser based on WordNet uses the synsets to find the word(s) maximizing Wu-Palmer similarity with the clue word.
- Codemaster based on a pre-trained embedding uses the embedding's vocabulary to find a clue word and a set of team words optimizing our custom distance function. Weights can be assigned to team, assassin, civilian and ennemy words. Codemaster based on WordNet work in a similar fashion by maximizing the Wu-Palmer similarity between the clue word and set of team words. Clue words are either taken from a list of 7,000 chosen words (version 2), or generated by looking for lowest common hypernyms between words (version 1).

In order to assess our bots' performance, the following setup has been designed:

1. Teams of bots, composed of a codemaster and a guesser, are defined. Seeds used to generate games are fixed.
2. For each seed, each team starts by confronting itself in a "mirror match".
3. For each seed, each team faces every other team twice, starting the same match as red and as blue in order to avoid potential bias due to unbalanced word grids.

When all games are over, valuable statistics such as win rates and average number of turns are collected to measure performances.

3 Results

Results presented in this report are based on 250 games simulated on 10 different seeds with 5 teams.

Basis	Type	Win%	Avg Turns
GloVe	Guesser	56%	8.04
W2V	Guesser	51%	8.48
WordNet	Guesser	46%	8.69
GloVe	CodeMaster	52%	8.33
W2V	CodeMaster	54%	8.49
WordNetV1	CodeMaster	29%	8.71
WordNetV2	CodeMaster	63%	8.83

Table 1: Bots marginal results

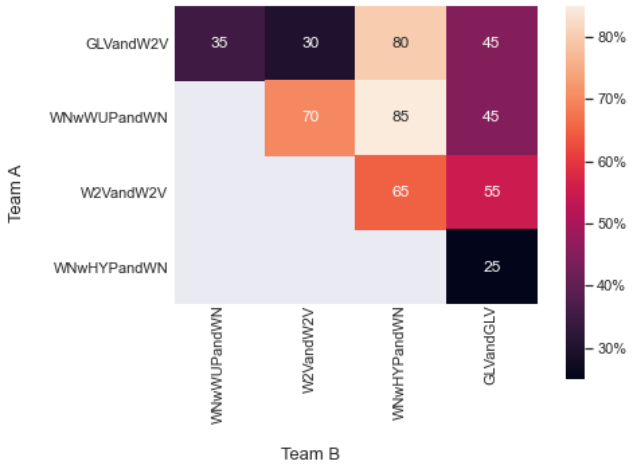
Table 1 and Table 2 present marginal win rates and average number of turns (on games where the assassin has not been called). Most of the games end between 7 and 9 turns, which means that each codemaster gave at least 4 words and that guessers guessed correctly around 1 or 2 words during each turn, which is an okay overall performance.

CodeMaster	Guesser	Win%	Avg Turns
GloVe	GloVe	56%	8.04
GloVe	W2V	48%	8.56
W2V	W2V	54%	8.49
WordNet1	WordNet	29%	8.71
WordNet2	WordNet	63%	8.83

Table 2: Teams marginal results

It seems rather clear that the first version of the WordNet codemaster performs badly compared to other algorithms. On the other hand, its second version is, at first sight, the best. Nevertheless, this second version is also the one associated to slower games, which is due to safer plays. GloVe and W2V-based CodeMasters and Guessers alike tend to give good enough results while not spending to much turns to reach victory. However, advanced analytics show that W2V-based algorithms tend to call assassins more often than GloVe-based bots, which makes them less reliable to play against humans.

Figure 1: Win rates of Team A over Team B



Competitive results displayed in Figure 1 show that the team entirely based on GloVe embedding is almost never defeated in the long run. The team based on WordNet and Wu-Palmer similarity (WordNet2) clearly beats every other competitors but struggles a little more against the full GloVe team. In the end, full GloVe team seems to be more fitted to play against humans as it adapts better to its opponents and it is faster than WordNet2.

4 Conclusion and Discussion

Using various NLP techniques such as word embeddings and synsets, bots that are able to play *Codenames* were implemented. The final bots make guesses and clues which generally make sense to humans. Benchmark results suggest that bots based on the use of hypernyms are the worst performing ones, whereas the best performing bots are based on GloVe embedding.

Several improvements may be added. Firstly, only five different teams were confronted in the benchmark due to excessive computing time otherwise. In addition, final bots are unable to modulate their answers depending on past guesses or clues from either team. Adding this feature would allow reinforcing the bots and making their behavior more human-like.

References

Adam Summerville. The codenames ai competition.

Kim Andrew et al. 2019. Cooperation and codenames: Understanding natural language processing via codenames. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.