Cooperation and Codenames: Understanding Natural Language Processing via Codenames

Andrew Kim, Maxim Ruzmaykin, Aaron Truong, Adam Summerville
California State Polytechnic University, Pomona
andrewhkim@cpp.edu, mruzmaykin@cpp.edu, aarontruong@cpp.edu, asummerville@cpp.edu

Abstract

Codenames – a board game by Vlaada Chvátil – is a game that requires deep, multi-modal language understanding. One player, the codemaster, gives a clue to another set of players, the guessers, and the guessers must determine which of 25 possible words on the board correspond to the clue. The nature of the game requires understanding language in a multi-modal manner – e.g., the clue 'cold' could refer to temperature or disease. The recently proposed Codenames AI Competition seeks to advance natural language processing, by using Codenames as a testbed for multi-modal language understanding. In this work, we evaluate a number of different natural language processing techniques (ranging from neural approaches to classical knowledge-base methods) in the context of the Codenames AI framework, attempting to determine how different approaches perform. The agents are evaluated when working with identical agents, as well as evaluated with all other approaches – i.e., when they have no knowledge about their partner.

Introduction

Games have been a popular test-bed for Artificial Intelligence since the inception of the field – with chess being used as a testbed for AI since 1957. Competitions based on games have largely been focused on games as an arena for adversarial competition – develop a bot that is able to beat the most other bots. Recently, there have been competitions devoted to cooperation in games, but this cooperation is typically not of a form like standard human cooperation. *Codenames* (by Vlaada Chvátil (Chvátil 2015)) is a game focused on natural language understanding – the board is composed of 25 words – and the goal is for a team of *codemaster/guesser(s)* to find all of the words associated with their team. The *codemaster* is given hidden information about the words on the board that are owned by their team, and it is up to produce single word clues so that their team of guessers can correctly identify their words. The clues must be semantically related to the words. The recently announced Codenames AI Competition (Adam Summerville 2019) uses Codenames as a testbed for natural language understanding. *Codenames* is unique amongst game AI competitions in that it requires:

1. Deep, multi-modal language understanding
2. Assymetric cooperation

Other game AI competitions have touched on these aspects – but not in concert – most notably, the Text-Based AI Competition(Atkinson et al. 2018) and The Hanabi Competition(Canaan et al. 2018). The Text-Based AI Competition presents a text-based adventure game environment for agents to explore – and potentially solve. This involves natural language understanding and common-sense reasoning not required by the other competitions. On the other hand, the Hanabi Competition is focused on cooperation with unknown agents in a game where players work towards a shared goal of beating the game.

This work uses the Codenames AI framework to assess the capabilities of different natural language processing approaches:

- WordNet – using a range of path similarity metrics
- word2vec
- GloVe

by assessing how well the different approaches work as both codemaster and guesser. This assessment looks at how well the bots do when paired with a *twin* – a bot using the same underlying technique – and with all non-twin partners as well. The codemasters are tested at a range of different of sensitivities, to see how well they trade off accuracy for speed. This work is a novel contribution to the fields of game playing and natural language understanding for

- the development of multiple Codenames AI bots for both the roles of *codemaster* and *guesser*
- the evaluation of how well different natural language processing cooperate with each other

In the rest of the paper we will first discuss how this work fits in to the fields of natural language understanding and game playing. Coupled with the rules of Codenames and the rules of the Codenames AI competition. We then describe the different techniques used and the structure of the bots. Finally we will assess the results of a round-robin tournament of all pairs of guessers and codemasters.

Related Work

Game competitions have been a popular test-bed for a variety of AI techniques. These have ranged from strategy games in the Starcraft AI Competition (Churchill 2018), to first-person shooters in the VisDoom competition (https://www.crowdai.org/challenges/vizdoom-2018 2018). More recently, there has been interest in games that require either textual understanding, cooperation, or epistemic reasoning – in the Text-Based AI Competition(Atkinson et al. 2018), The Hanabi Competition(Canaan et al. 2018), and One Night Ultimate Werewolf (Eger and Martens 2018) – respectively. The recently announced *Codenames AI competition* requires textual understanding, cooperation, *and* epestemic reasoning. The codemaster and guesser need to understand the words being presented to them, they need to work towards their common goal, and – perhaps most importantly – they need to understand what their partner understands, so that they can successfully cooperate. Work by Bard et al. used reinforcement learning to train Hanabi playing bots (Bard et al. 2019). They found that bots that had a paired partner were able to play nearly perfectly, but when paired with a partner that did not have the same communication strategy the bots performed horribly, with nearly no chance of winning – demonstrating the challenge of working with unknown teammates when no external communication is allowed.

While, to our knowledge, this work is the first to compare different word similarity and word embedding approaches in the context of games (and more specifically, *Codenames*) – this is not the first work to compare them in other natural language processing contexts. Naili et al. compared Word2Vec, GloVe, and LSA vectorial semantic approaches in the domain of topic segmentation – finding that both GloVe and Word2Vec outperform LSA (Naili, Chaibi, and Ghezala 2017). Rücklé et al. compared GloVe, Word2Vec, and concatenated combinations thereof in a number of different contexts – Argumentation Mining, Sentiment Classification, Opinion Polarity, and Question-Type Classification. They found that concatenations of GloVe and Word2Vec worked the best – meaning that while the approaches are similar – they each have relative merits that can improve upon each other (Rücklé et al. 2018). In his masters thesis Handler uses WordNet to assess the types of relationships found by word2vec – finding that word2vec finds more similarity between synonyms, hypernyms, and hyponyms – ahead of meronyms and holonyms (e.g. it finds more similarity between ‘wheels’ and ‘tires’ than ‘wheels’ and ‘car’). However, none of these approaches assess how well the different approaches agree with each other in terms of similarity – which our work addresses.

Codenames

Codenames is an asymmetric cooperative board game where a team – composed of a codemaster and some number of guessers – work to find their hidden words as quickly as possible. Table 1 shows an example board from the game. The colors represent the hidden information that the codemasters are privy to:

DAY	SLIP	*SPINE*	**WAR**	*CHICK*
HAND	WALL	**AMAZON**	DEGREE	**GIANT**
CLOAK	STREAM	*CHEST*	**HAM**	*DOG*
FALL	**CENTAUR**	EMBASSY	GRASS	FLY
OIL	*COLD*	*HOSPITAL*	**MARBLE**	CAPITAL

Table 1: An example Codenames board. **Bold+Red words** are those for the red team, *Italic+Blue words* are for the blue team, gray words are civilians, and the remaining word is the assassin.

- Gray – A civilian – an unaffiliated card that ends the turn when selected by a guesser
- Black – The assassin – an unaffiliated card that instantly causes the guessing team to lose the game
- Blue/Red – The secret agents – the affiliated cards that the teams are trying to select

The game has the following structure:

1. The team with the most words goes first.
2. The codemaster of the current team gives their clue – a single word and a number.
3. The guessers of the current team select a clue – if at any time they choose a card that does not belong to their team, their turn ends.
4. The guessers can choose to repeat **3.** up to the given number plus 1.
5. Play passes to the opposing team and the game goes to **2.**

The rules of the game disallow the codemaster from saying more than a single word as a clue, saying a clue that is the word itself, and a clue that is too similar to a word on the board.

For example, the red team would start in the example setup of table 1. The codemaster for the red team might wish to use a clue to link **WATER** and **AMAZON** – perhaps “river” or “basin” – but the assassin word **STREAM** confounds that plan – so perhaps they choose “ARCHER 3” – hoping their guessers select **WAR**, **AMAZON**, and **CENTAUR**. This demonstrates one of the key challenges of Codenames – most of the words have multiple meanings – as evidenced by the above example. E.g., **AMAZON** could refer to the river, the company, or the mythical civilization.

While Codenames is a competitive game, it can also be played in a solitaire style, which is the basis for the competition. The competition has a team of bots competing to complete the board in as few turns as possible. There is no opposing team, with the game ending in one of two states:

1. Win – The team successfully clears the board – their score being the number of turns it took to complete the board
2. Lose – The team incorrectly selects the assassin or chooses all of the opposing team’s words before their own, their score will be 25

The competition allows for competitors to enter either a guesser, a codemaster, or both. The competition has three tracks:

- Paired Guesser/Codemaster

- Solo Guesser
- Solo Codemaster

If a competitor enters both a guesser and codemaster, they can compete in the paired track, where their guesser will be paired with their codemaster, and they will compete against other such teams.

Furthermore, the guesser and codemaster bots are also be entered into the solo competition where they will be paired with other entrants, seeing how well they can communicate and cooperate when they do not know their partner.

In this work, we analyze different natural language processing techniques in the context of the Codenames AI competition – seeing how well paired codemasters and guessers can perform, as well as how non-paired teams perform. The following section details the overall approach for the bots, as well as the different natural language processing techniques.

Bot Methodology

All of the bots used in this analysis follow the same basic approach – respective of the role of codemaster or guesser – , regardless of the natural language processing technique used. Pseudocode for the codemaster is:

Algorithm 1 Codemaster

```
1:  procedure ProduceClue
2:      R ← []
3:      for r_w ∈ red_words do
4:          for w ∈ words do
5:              R[r_w][w] ← $D$(r_w,w)
6:      B ← []
7:      for b_w ∈ bad_words do
8:          for w ∈ words do
9:              B[b_w][w] ← $D$(b_w,w)
10:     $C_i \leftarrow 0$
11:     best ← null
12:     $d \leftarrow \infty$
13:     for $i \in (1..|$red_words$|)$ do
14:         for $r_c \in$ Combination(red_words, $i$) do
15:             for w ∈ words do
16:                 $w_d \leftarrow \infty$
17:                 for b_w ∈ bad_words do
18:                     if B[b_w][w] $< w_d$ then
19:                         $w_d \leftarrow$ B[b_w][w]
20:                 $d_r \leftarrow 0$
21:                 for r_w ∈ $r_c$ do
22:                     if $R[r_w][w] > d_r$ then
23:                         $d_r \leftarrow R[r_w][w]$
24:                 if $d_r < d$,
25:                   $d_r < w_d$
26:                   $d_r < t$ then
27:                     $d \leftarrow d_r$
28:                     best ← w
29:                     $C_i \leftarrow i$
30:     return best, $C_i$
```

In lay terms, the bot looks through all combinations of red words from size 1 up to the number of the remaining red words. The bot chooses the best word (assuming the supplied distance function D supplies a lower distance for more similar words) subject to:

- The distance between the clue and any of the "bad words" (i.e. the blue, grey, or black words) is further than the clue and the worst of the red target words
- The distance between the worst red target word is smaller than the supplied threshold, t

The supplied threshold t (*low*$= 0.3$, *medium*$= 0.5$, *high*$= 0.7$) limits how aggressive the bot is. The words come from a list of the 10,000 most common English words, with all words that lemmatize to the same word condensed into the lemmatized form (e.g. rub, rubs, rubbing, etc. all lemmatize to rub).

The guesser is much simpler than the codemaster and works as follows:

Algorithm 2 Guesser

```
1:  procedure MakeGuess(clue)
2:      best ← null
3:      $d \leftarrow \infty$
4:      for word ∈ board do
5:          if $D$(word,clue) ¡ $d$ then
6:              $d \leftarrow D$(word,clue)
7:              best ← word
8:      return best
```

Simply speaking, it looks over all of the words on the board, and selects the one that most closely matches the clue.

While the broad bot infrastructure is the same for all of the bots, the internal workings for $D(w_1, w_2)$ differ for the bots. The following sections detail how the different natural language techniques operate.

WordNet

WordNet is a venerable Natural Language Processing framework that represents words in a graphical structure (Miller 1995). All words in WordNet are composed of *synsets* (short for synonym sets) – a set of meanings that the word might take on. E.g., *dog* has the synsets for referring to a canine, a hotdog, the action of chasing, and colloquialisms ("you dirty dog", "you lucky dog", etc.). These synsets then relate to each other through four different types of semantic relationships:

1. Hypernomy – A hypernym of a word is one that is more general – Animal is a hypernym of Dog
2. Hyponomy – A hyponym of a word is one that is less general – Poodle is a hyponym of Dog
3. Meronomy – A meronym of a word is one that the word is a part of – Tail is a meronym of Dog
4. Holonomy – A holonym of a word is one that contains the accommodated word – Animal Kingdom is a holonym of Dog

These relationships can then be traversed to find the similarity between two words; however, a number of different traversals are possible. For this work we considerred:

Path Similarity – A score for how similar two word senses are, based on the shortest path that connects the synsets in the (hypernym/hypnoym) taxonomy defined as:

$\frac{1}{distance+1}$

Leacock-Chodorow Similarity – A score that uses the shortest path as in **Path Similarity** and the maximum depth of the synsets in the taxonomy, defined as (Leacock and Chodorow 1998):

$-\log(\frac{p}{2d})$

where p is the shortest path length and d the taxonomy depth.

Wu-Palmer Similarity – A score based on the depth of the two synsets in the taxonmy and their deepest ancestor node – their Least Common Subsumer (LCS) – defined as (Wu and Palmer 1994):

$2 * \frac{depth(lcs)}{depth(s_1)+depth(s_2)}$

Resnik Similarity – A score based on the Information Content (IC) of the LCS. The IC is defined as (Resnik 1999):

$-\log(p_x)$

Where p_x is the probability that a randomly selected word from a corpus enacts a concept in WordNet (e.g. the word 'dog' counts for the words 'dog', 'mammal', 'animal', 'organism', and 'entity'). For this work, we used the Brown corpus for all IC related measures.

Jiang-Conrath Similarity – A score based on the IC of the LCS and that of the two input synsets, defined as:

$\frac{1}{(IC(s_1)+IC(s_2)-2IC(lcs))}$

Lin Similarity – A score based on the IC of the LCS and that of the two input synsets, defined as (Lin 1998):

$\frac{2IC(lcs)}{IC(s_1)+IC(s_2))}$

As a note, all of these measures are defined such that more similar words have a higher similarity. Since the pseudocode shown before assumes that the similarity measures are distances (i.e., lower being better), it is necessary to invert the measures (or flip the signs of all comparisons). For this work, the Natural Language Toolkit (NLTK) was used to access the WordNet database (Bird 2006).

Vectorial Semantics

Vectorial semantics are an NLP approach that relies on embedding words in a high (typically 100+) dimensional vector space. These approaches can come to these word vectors in different ways, but once the word vectors have been found, they can be used in similar ways. Most notably, vector math operations can lead to different semantic operations – e.g., $A - B + C = D$ enacts "A is to B as C is to D", i.e., analogy. Similarity for vectorial semantics operates by finding the word that is closest given some distance metric for the vector space. There are infinitely many possible distance metrics that could be used, ranging from the Euclidean L_2 or Manhattan L_1 norm to more exotic norms, but the most commonly used distance is the cosine distance, e.g. the cosine of the angle between the two vectors. This is most commonly defined as:

$1 - \frac{w_1 \cdot w_2}{||w_1||_2||w_2||_2}$

I.e., One minus the dot product of the two vectors, after normalization, the measure is 0 for identical vectors, 1 for orthogonal vectors, and 2 for exactly opposing vectors.

Of course, as mentioned above, while the process of finding the similarity using word vectors is the same once the word vectors have been derived, the details of how to get the word vectors have specific differences. For this work, we considered two different word vector approaches – word2vec and GloVe – which we will now discuss.

Word2Vec

Word2Vec (Mikolov et al. 2013) is a commonly used word vector training approach that actually comprises two different approaches – the Continuous Bag Of Words (CBOW) and the Skip-Gram models. The two approaches use a similar idea, given a window in a text – use a word and its context to train a neural network – the weights of which give the word vectors. E.g., given the sentence "The quick brown fox jumped" the word might be "brown" with the corresponding context ["The", "quick", "fox", "jumped"]. The difference between the CBOW and Skip-Gram models come from how the word and contexts are used – the CBOW model takes the context as the input and tries to predict the word, while the Skip-Gram model takes the word and tries to predict the context. For this work, we used a 300 dimensional pre-trained Skip-Gram model trained on the Google News corpus.

GloVe

"**Glo**bal **Ve**ctors for Word Representation" (GloVe) is another approach for deriving word vectors from a corpus (Pennington, Socher, and Manning 2014). The key insight to GloVe is that probability ratios for word carry the type of information that is desired from vectorial semantics. E.g., given Pr(x|'ice')/Pr(x|'steam') the ratio is high for words more associated with ice like 'solid', very low for words more associated with steam like 'gas', close to 1 for unrelated words like 'fashion', and close to 1 for similarly related words like 'water.' Given this insight, GloVe is trained by a linear regression that tries to learn weights such that the weights associated with a word try to predict the log of the co-occurrence counts of the word and its contexts:

$\sum_{i,j=1}^{V}(w_i + b_i - logX_{ij})^2$

where w, b are the weights and bias associated with word i, X_{ij} is the co-occurrence count for word i and word j, and V is the size of the vocabulary. For this work we considered the 50, 100, 200, and 300 dimensional GloVe embeddings.

As discussed in the related work, it has been demonstrated that in certain contexts the concatenation of word2vec and GloVe vectors can improve performance (Rücklé et al. 2018), so we too consider them.

Results

To assess the capabilities of the different NLP techniques, we ran a round-robin tournament of 30 games for all pairs of codemaster and guesser (the same 30 boards were presented to all pairs). As a note, we found that the WordNet based codemasters were both extremely time inefficient and mostly inscrutable to us for the chosen clues, so we did not consider them in the tournament – perhaps in future work they can be more fully assessed. For each of the codemasters we considered 3 different threshold levels, to see how the

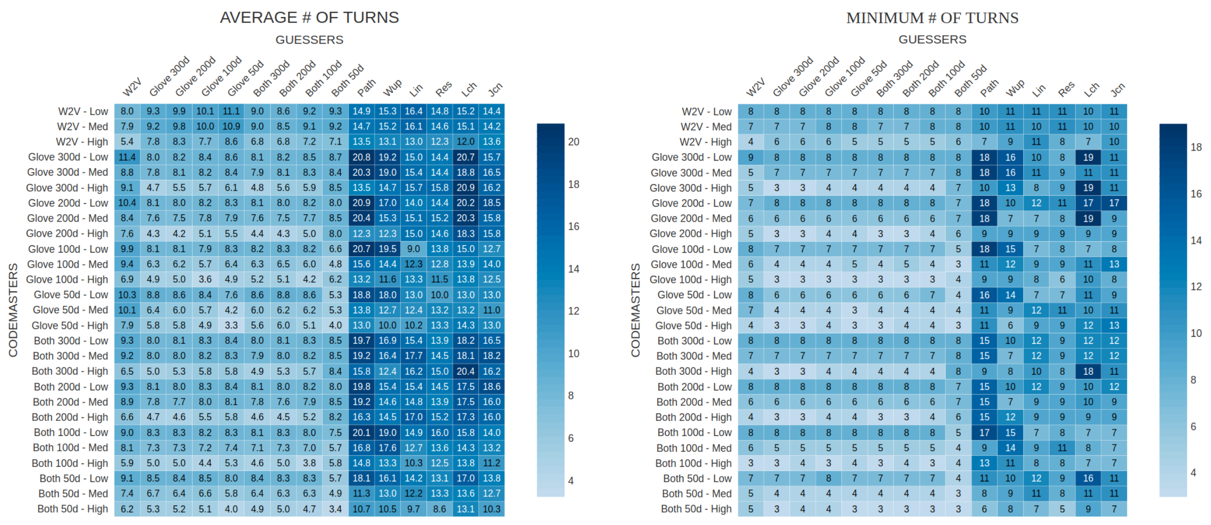

Figure 1: The average (left) and minimum (right) number of turns that the different pairs take to finish the game. These turn counts are based only on games that are won, ignoring the games where the team loses. Lighter cells are better than darker. We see that the more aggressive thresholds lead to better min and average turn counts – with 3 turn wins not being uncommon. The more conservative thresholds lead to 7 or 8 turn games – 1 clue per turn. We notice that while the various vectorial semantic approaches work well together, the WordNet based approaches universally do not play well with vectorial semantic approaches.

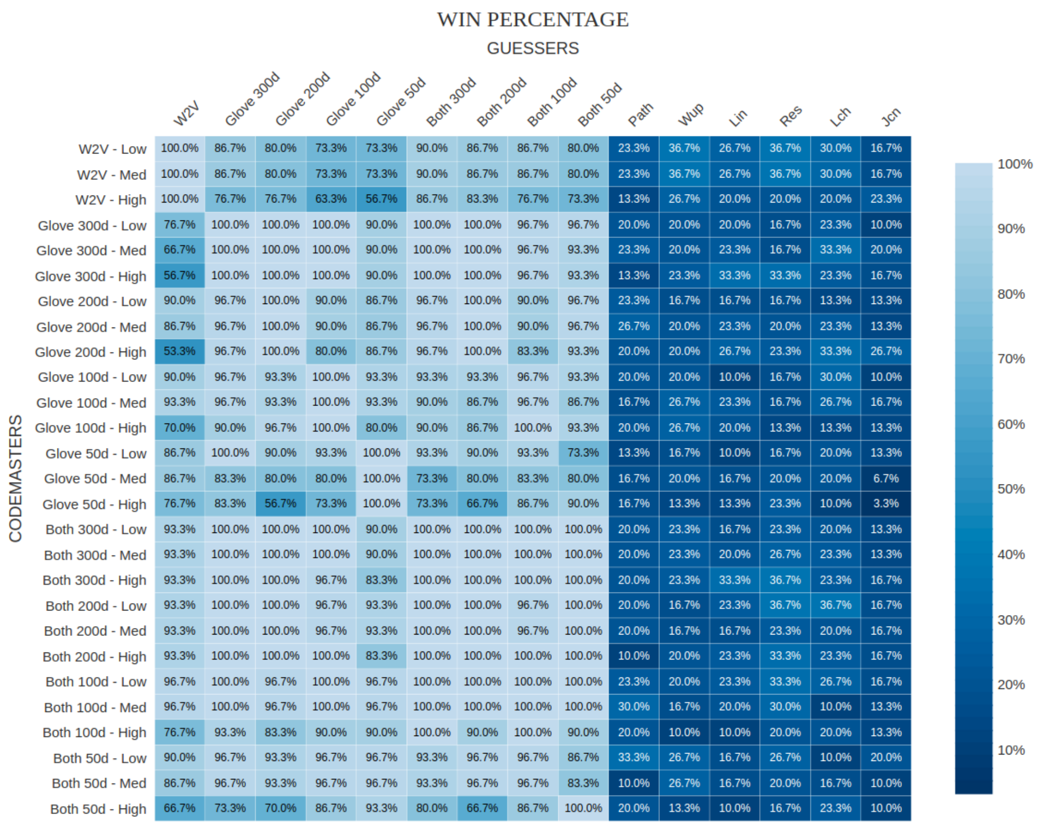

Figure 2: The win percentages across the tournament for the codemaster-guesser pairs. The paired *twins* have a 100% win rate, as expected. Also, as the threshold increases the win rate decreases – again, as expected as the lower threshold means a more conservative/less aggressive codemaster, one that is more likely to win (at the cost of taking more turns). Universally, the WordNet based guessers do very poorly – never breaking an above 50% win rate.

different approaches trade off speed and accuracy – a lower threshold being more conservative. The number of turns, average and minimum, can be seen in figure 1, while the win rates can be seen in figure 2.

Unsurprisingly, the codemasters that are paired with their corresponding guesser achieved a 100% win rate. Somewhat interestingly, we see that word2vec codemasters do worse as the dimensionality of the GloVe based guessers increases, and vice-versa. Seemingly, low-dimensional GloVe corresponds to high (300) dimensional word2vec – which is something that, to our knowledge, has not been observed elsewhere. Concatenating word2vec and GloVe results in the best results across the board – the concatenated word2vec and 300d GloVe has $\geq$ 90% win rate across the vectorial semantic based approaches (higher than all other tested codemasters). However, the vectorial approaches do very poorly with the WordNet based approaches – topping out at a 36.67% win rate and dipping to a 3.3% win rate. While this study did not have a human component, our personal analysis found that the WordNet based approaches tended to choose clues in ways that were hard to justify, while the vector based approaches were often more reasonable. This is obviously not hard, obviously demonstrated proof, but the overall agreement between the vector based approaches in contrast to the WordNet demonstrated approaches would seem to favor vectorial machine-learned approaches.

The turn rate results only show goes where the team won. As with win-rate, the concatenated word2vec + GloVe performed best here, with the 300d combined with Word2Vec averaging a 6.9 turns needed to win any given game. Unsurprisingly, the more aggressive thresholds led to faster win times. In contrast to the win-rate results, the lower dimensional GloVe vectors required less turns than that of higher dimensional GloVe vectors when concatenated with word2vec – for the Codemasters. Even more interestingly, the opposite effect is found across the guessers, with the higher dimensional GloVe-concatenated guessers performing better than the lower dimensional concatenated guessers. Perhaps this is due to the seeming result that low dimensional GloVe corresponds well with word2vec – leading to games where the 2 approaches are simpatico, but further study is required to determine exactly why the higher dimensional GloVe results are more stable, yet less optimal, than the lower dimensional GloVe vectors.

Looking at the minimum number of turns, the minimum across all tested guessers and codemasters was 3 turns – averaging $2\frac{2}{3}$ words a turn – which to the authors seems to be an unamazing, yet very good game for Codenames. The concatenated (low-dimensional) codemaster averaged 4.7 minimum turn games across all guessers – and on average a 7.1 turn game – a full turn faster than the next closest codemasters (the 100d GloVe and 100d Glove + word2vec at 8.1 turn games). Interestingly, the low dimensional, high-threshold concatenated codemaster performed decently with the WordNet based guessers. In the best-case – an average 7 turn game across the WordNet guessers. That being said, the average case for the WordNet guessers was at best 10.5 turns – meaning that they missed at least 2 on average.

Obviously, the bots that are epistemically paired (i.e. the *twin* codemaster-guesser pairs) perform the best, with 100% win rates and low (3.2 turn) number of turns required to find all words. Across the field, we see that concatenating word2vec and GloVe performs better than either on their own; However, the dimensionality of the GloVe vectors used has different tradeoffs – high-dimensional codemasters have the best win-rate, while low-dimensional codemasters have the best win times. Interestingly, for the guessers increasing dimensionality leads to both higher win-rates and lower number of turns. This would indicate that a competitor in the competition would do best to consider a mixed codemaster-guesser team to pair best across the field.

Conclusion and Future Work

In this paper we have detailed two different bot frameworks for the Codenames AI competition – a *codemaster* and a *guesser* – and have analyzed these both with a wide variety of Natural Language Processing backends, ranging from classical knowledge-base approaches deriving from WordNet and from modern machine-learned vectorial semantics approaches, word2vec and GloVe. We analyzed these frameworks across a round-robin tournament, pairing all codemasters (with a number of different aggressiveness thresholds) and all guessers – to see how well these different approaches can cooperate with each other. We found that the vectorial semantics based approaches universally worked well with each other – win rates ¿80% on average – while the WordNet based approachs universally performed poorly with the vector based approaches – win rates ¡40%, and that concatenating word2vec and GloVe vectors has the best overall results – working well with stand-alone word2vec, stand-alone GloVe, and concatenated approaches. Interestingly, we found that increasing the dimensionality of the codemaster leads to more wins – at the cost of speed. Conversely, we found that increasing the dimensionality of the guesser leads to *both* increased win percentage and increased speed – leading us to believe that Codenames' AI entrants utilizing these approaches should have a mixed guesser-codemaster team, in terms of dimensionality.

In the future, we would like to examine more recent vectorial semantics approaches – namely, ELMO (Peters et al. 2018) and BERT (Devlin et al. 2019). Both create contextual word vectors albeit via different approaches – ELMO uses a Long Short Term Memory Recurrent Neural Network (LSTM RNN) and BERT is instead based off of the Transformer (Vaswani et al. 2017). Given the multi-modal word meanings inherent in Codenames – contextual word vectors would seem to be a powerful tool. These two approaches could lead to more nuanced codemasters and guessers – more in line with human reasoning. Along those lines, we would like to conduct a human study to assess which of the approaches best align with humans. Finally, these bots are very simplistic in their play – a more nuanced approach would pay attention to how their partners respond.

References

Adam Summerville, Andrew Kim, M. R. A. T. 2019. Codenames AI Competition [Accessed on June 10, 2019]. *https://sites.google.com/view/the-codenames-ai-competition*.

Atkinson, T.; Baier, H.; Copplestone, T.; Devlin, S.; and Swan, J. 2018. The Text-Based Adventure AI Competition. *arXiv preprint arXiv:1808.01262*.

Bard, N.; Foerster, J. N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H. F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; et al. 2019. The hanabi challenge: A new frontier for ai research. *arXiv preprint arXiv:1902.00506*.

Bird, S. 2006. Nltk: The natural language toolkit. In *COLING ACL 2006*, 69.

Canaan, R.; Shen, H.; Torrado, R.; Togelius, J.; Nealen, A.; and Menzel, S. 2018. Evolving agents for the hanabi 2018 cig competition. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Churchill, D. 2018. AIIDE StarCraft AI Competition [Accessed on June 10, 2019]. *https://www.cs.mun.ca/ dchurchill/starcraftaicomp/*.

Chvátil, V. 2015. Codenames. *Czech Games*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*.

Eger, M., and Martens, C. 2018. Keeping the story straight: A comparison of commitment strategies for a social deduction game. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

https://www.crowdai.org/challenges/vizdoom-2018. 2018. VizDOOM Competition [Accessed on June 10, 2019].

Leacock, C., and Chodorow, M. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database* 49(2):265–283.

Lin, D. 1998. An information-theoretic definition of similarity. *International Conference on Machine Learning* 98.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Naili, M.; Chaibi, A. H.; and Ghezala, H. H. B. 2017. Comparative study of word embedding methods in topic segmentation. *Procedia computer science* 112:340–349.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Resnik, P. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research* 11:95–130.

Rücklé, A.; Eger, S.; Peyrard, M.; and Gurevych, I. 2018. Concatenated power mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wu, Z., and Palmer, M. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 133–138. Association for Computational Linguistics.