5-2020

Probing NLP Conceptual Relatedness Judgments Through the Word-Based Board Game Codenames

Tracey Mills
Dartmouth College

PROBING NLP CONCEPTUAL RELATEDNESS JUDGEMENTS THROUGH THE WORD-BASED BOARD GAME CODENAMES

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Computer Science

by

Tracey Mills

DARTMOUTH COLLEGE

Hanover, New Hampshire

May 2022

Examining Committee:

Jonathan Phillips and Soroush Vosoughi, Chair

Alberto Quattrini Li

Rolando Coto Solano

Dean of the Guarini School of Graduate and Advanced Studies

Acknowledgements

I am grateful to my advisors for their guidance throughout the completion of this thesis. First, I would like to thank Professor Jonathan Phillips for wondering 2 years ago how good we could make AI at playing Codenames, and from then on for his constant support of my work on the project. I could not have completed this thesis without his help in asking interesting questions, analyzing findings in response to these questions, and situating them in their broader philosophical context. I would also like to thank Professor Soroush Vosoughi who agreed to be my co-advisor for this project despite already being over capacity, and offered valuable insights, directions for analysis, and general guidance on the process of completing an honors thesis, while also helping me to clarify the big-picture goals of my work.

Abstract

In this paper I evaluate the ability of different Natural Language Processing (NLP) techniques to make human-like word relatedness judgements in a variant of the word-based board game Codenames. I analyze a variety of statistical and knowledge based approaches, combinations of these, and techniques for incorporating the wider game context into relatedness judgements. While no approach explored here reaches human performance, simple word embedding based approaches incorporate a surprising amount of the useful information captured by other techniques. I attempt to characterize the limitations of these approaches in relation to human game play, although differences are largely not systematic. Finally, I discuss these results in terms of future directions for the field of NLP.

Contents

Chapter 1

Introduction

Codenames, decribed in detail below, is a game that hinges on determining the relatedness of words. Such a task is simple to describe and in principle easy for humans. It is therefore an interesting question whether or not current Natural Language Processing techniques, whose success is accelerating in tasks such as translation, text generation, analogy tasks, and many others [26], can achieve human-like performance on this task. Since a model with a human-like theory of how concepts relate to words should be able to determine word relatedness in a human-like way, comparing a model's performance on this task to human performance will shed light on the extent to which these models are learning human-like language representations.

Section 1.1

Codenames

1.1.1. Basic game play

Codenames is an award-winning word-based boardgame designed by Vlaada Chvátil (see https://czechgames.com/en/codenames/). It is played by two teams of two, where each teammate has a well-defined and perhaps deceptively simple role. The

board consists of 25 cards each with a single word written on it, with either 8 or 9 words assigned to each team, 7 of the remaining words neutral and assigned to neither team, and 1 word representing the "assassin." In each team, one partner is the spymaster, and one partner is the guesser. The spymaster knows which cards are assigned to their team, which are assigned to the other team, and which are neutral. Their task is to give one-word clues to their partner that will help them guess some of their team's words, while not guessing the other words. Teams alternate turns, in which the spymaster gives a clue along with the number of words that the clue applies to (usually 2-4 words in personal experience). The guesser then attempts to guess the words intended by the spymaster. They give their guesses one at a time, with each guessed word being removed from the game. The guesser can continue guessing as long as they choose their own team's words; however, if they guess a neutral word or a word belonging to the other team, these words are removed from the game and the turn ends. If the guesser chooses the assassin word, they immediately lose. Apart from this special case, the winning team is the first to have all of their assigned words guessed by either team's guesser and removed from the game.

1.1.2. Nature of clues

The task of the spymaster is highly creative, with the only restrictions on clues being that that they are either single words or proper nouns, and that they do not contain or appear in any of the words on the board. Not only can clues relate to the board words in any way, but almost all of the 400 possible board words from the original version of Codenames are polysemous, with more than one possible meaning (e.g. "BANK," "PUPIL," "ROCK," etc.). Within this large space of possible board word meanings, and clues which may bear any sort of relationship to any of these meanings, successful teams must somehow have a joint understanding of what these word meanings are, which bear what sorts of relationships to a given clue, and precisely the relative

Figure 1.1: This snapshot of Codenames play, captured from https://czechgames.com/en/codenames/, shows an example board of 25 words, along with the square card which is in view of each team's spymaster and shows spatially which words belong to each team (red or blue), which words are neutral (beige), and which word represents the assassin (black).

strength of these relationships.

Section 1.2

Problem Statement

In this paper I hope to characterize the ability of various NLP techniques to achieve human-like performance on the word-game Codenames, with the joint goals of better understanding how humans determine word relatedness and in what ways current NLP approaches are equipped to do so in a similar way.

Section 1.3

Paper Organization

I begin this paper by describing a simplified Codenames task, and reporting human performance on this task. I next describe and evaluate the performance of various

NLP approaches on this task: naive statistical approaches, statistical approaches which incorporate contextual information, knowledge-based approaches, combinations of knowledge-based and statistical approaches, and multi-modal techniques. I then more specifically characterize the limitations of these approaches in relation to human performance, and close by discussing implications of these results for the field of NLP and potential future directions.

Chapter 2

Related Work

This paper aims to build off of prior work investigating the use of NLP to approximate human performance on Codenames-inspired word games.

Section 2.1

AI Performance in Codenames

Some prior work has investigated the performance of different NLP approaches in bots that play with and against each other. Kim et al. analysed the performance of embedding-based (Word2Vec and GloVe) and knowledge-based (WordNet) spymasters and guessers when paired with each other [11]. They found the embedding-based approaches to be the most successful, with embedding-based spymasters performing better when paired with an embedding-based guessers than when paired with the WordNet-based guesser, and approaches using concatanated GloVe and Word2Vec embeddings having the most success over all. Additionally, the authors described clues given by the WordNet-based spymaster as "inscrutable" and "hard to justify."

Jaramillo et al. followed up on this work by comparing different approaches to the winning concatenated Word2Vec and GloVe approach [9]. Using dictionary definitions and Wikipedia summaries for the board words as a corpus, they used both

a Term-Frequency Inverse Document Frequency algorithm, which determined word-relatedness based on a word's frequency in the context of a given word relative to its overall frequency, and a Naive Bayes algorithm which determined each of the board word's probability of belonging to each of 118 classes (e.g. "animals", "countries", "food", etc), where the class that a word most probably belongs to would be deemed a good clue for that word. Neither of these approaches neared the performance of the embedding-based approach implemented by Kim et al. However, Jaramillo et al. also implemented an embedding-based approach using word embeddings from the large-scale transformer-based language model GPT-2. This approach performed competitively to the winning approach of Kim et al. when paired with other bots, and outperformed it on average when paired with each of 10 human players, tentatively suggesting that GPT-2 embeddings capture more human-like representations of words than the combined Glove and Word2Vec embeddings.

Overall, these sets of results speak to the relative compatibility of different embedding-based approaches, while also perhaps expressing pessimism for the use of knowledge graphs such as WordNet for the task. However, they do not tell us how these different approaches compare to human performance.

Section 2.2

AI and Human Partnerships

Other work has been done to test the compatibility of such approaches with human players. Koyyalagunta et al. implement knowledge and embedding-based AI spymasters which give clues to human guessers [13]. Their knowledge-based spymaster is implemented with BabelNet, in which edges between words are labeled with the relationship between those words. The authors found that clue quality improved when they restricted a word's nearest neighbors to those that were only one edge away, or

had only hypernym relationships for each edge after the first. With this specially constructed graph, they found the knowledge-based spymaster had similar success to the vector-based approaches, while being more transparent. Additionally, they found that both types of spymasters improved when they avoided especially common or especially uncommon clues, and when they incorporated word embeddings built from dictionary definitions into their scoring function. Each spymaster gave clues for 2 words on a 20 word board, and for the most successful spymaster, which used the described biases along with fast text embeddings, people guessed on average 66.7% of these words correctly in the first 2 tries. These results certainly do that these approaches are compatible with human play, although Koyyalagunta et al. do not provide comparable data for the success rate of human spymasters.

Section 2.3

Emulating Human Performance

Shen et al. (2018) and Kumar et al. (2021) use similar and other approaches to model human performance directly [27, 14]. Shen et al. gathered data on people playing a modified version of Codenames, in which the spymaster chooses between a set of adjectives to describe a subset of possible nouns, and guessers guess the nouns that are intended to be described by the spymaster's chosen adjective. They found that bigram co-occurence frequencies outperform embedding-based and knowledge-based approaches in modeling both human spymasters and guessers. Bigrams are especially suited to modelling human behavior in this particular version of the game, as adjectives and the nouns they describe tend to occur consecutively. In the original paradigm, in which clues are not restricted to being adjectives that describe the intended word but can be of any type (i.e. synonyms), clues and intended words may not co-occur consecutively to the same extent. Regardless, this finding picks out bigrams

as effective models for at least one strategy employed by humans in Codenames-like tasks. Shen et al. additionally investigate potential pragmatic aspects of human game-play by evaluating responses in terms of a pragmatic reasoning framework in which spymasters and guessers are modeled as recursively reasoning about the other's knowledge and likely inferences. Interestingly, they found the literal, non-pragmatic model to be a better fit, suggesting that spymasters and guessers do not recursively reason about each other in this manner.

Kumar et al. designed a version of Codenames in which spymasters are asked to come up with a clue for 2 specified words on a 20 word boar, and guessers try to guess these 2 words given the spymaster's clue. They compared the ability of embedding-based and associative models to predict human play. Associative models are built by asking human participants what words come to mind in response to a given cue word, and keeping track of the frequency of each response. In theory, these models should be expected to outperform embedding-based approaches, since this task is similar to the Codenames task in which spymasters pick clues that are associated with words on the board, and guessers in turn pick words that are associated with the given clue. Kumar et al. did indeed find associative models to outperform embedding-based models, with the most successful associative model, based on the Small World of Words word association database [3], correctly predicting 60.33% of human guesses and 21.67% of clues, and GloVe and Word2Vec based models correctly predicting 39.69% of guesses and 8.33% of clues, and 41.11% of guesses and 3.33% of clues respectively. While the associative model clearly captures aspects of word relations that embedding-based models miss out on, these models are still far from perfectly emulating human performance.

The authors further characterized what information is captured by each model by labeling what type of relationship clues had with intended words on each trial and

analyzing how well the models predicted each type of clue. They found that both model types were best at predicting coordinate clues, which are clues that belong to the same category as at least one of the intended words (i.e. apartment, villa). They also found a larger difference in predictive accuracy between associative and embedding-based models for coordinate clues and hierarchical clues, which group the intended words into a superordinate category, than for locative clues which highlight a location based aspect of an intended word, or attributive clues which describe a property or feature of an intended word. Finally, embedding-based models had the least success predicting hierarchical clues, suggesting that such structured conceptual relationships between words are not well-captured by these word embeddings.

The work I have discussed establishes a general consensus that no one currently existing NLP approach achieves human-like performance in Codenames. Embedding-based approaches seem to have the most potential for this goal, and benefit from the integration of more structured conceptual information, such as adjective-noun co-occurrence frequency, dictionary definitions, and hierarchical relationships which might be captured by knowledge graphs. I hope to build off of prior work by incorporating new, state of the art approaches such as GPT-3 text prediction and combining approaches in order to maximize their strengths. I also characterize and address some of the theoretical limitations faced by various NLP approaches in determining word relatedness.

Chapter 3

Human Performance

In this chapter I describe human performance on the Codenames task. Although both the guesser and spymaster role present interesting challenges for NLP, I chose to focus my analysis on the guessing portion of the task. While this may initially seem to be the "easier" problem, since it involves choosing from set words rather than coming up with a clue from an infinite option space, I argue that it is the more difficult and more interesting problem to solve with NLP given this paper's goals. While spymasters need draw only on their own knowledge to come up with a clue, successful guessers might experience clues that are outside their knowledge domain. Thus the guessing task better reveals deficiencies in the information captured by different approaches. The most notable simplifications of this task from the original Codenames guessing task are that the board is reduced to 12 words, spymasters must give a clue for exactly 3 words, there is no opposing team, and only one round is played per board. Guessers also do not need to pick words in order, and so they do not stop guessing once they have guessed incorrectly. These simplifications were made to isolate the task of drawing relationships between words from other strategic aspects of game play.

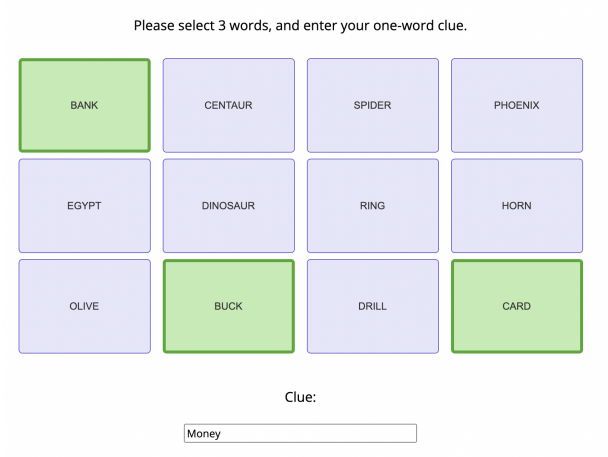

Figure 3.1: Each participant in study 1 saw this example before beginning the task, along with the following text: "In the example below, the clue money applies to the selected words bank, buck, and card. Money is kept in banks, a buck is a slang term for money, and people use cards to pay for things in place of money. While the other words may also bear some relation to the clue money, given this clue, it seems likely that someone would pick the three selected words instead of any of the others."

Section 3.1

Methods

3.1.1. Study 1

The purpose of study 1 was to gather data on human spymasters. I first constructed 400 12-word boards by randomly selecting words from the 400 cards in the original Codenames game. Participants were then recruited from Prolific ($N = 100$, $M_{age} = 33.45$, $SD_{age} = 13.77$, 47 females, 2 other) to act as spymasters for these boards. Participants were instructed to select 3 words from the board of 12, and give a one-word clue that would allow another person to guess those 3 words if given the same board. Each participant completed 10 trials, giving a clue for a different board on each trial. They were shown the example board in Figure 3.1 before beginning the task.

After completing 10 trials, participants saw the same boards, along with the clues

they gave, and were asked to re-select the words they had in mind for that clue. Trials in which participants did not correctly re-select all 3 words, or in which they gave invalid clues (more than 1 word or containing or appearing in a board word) were excluded from subsequent analyses, so that 135 trials were excluded out of a total 1036.

3.1.2. Study 2

The purpose of study 2 was to gather data on human guessers. From study 1, I collected 901 unique valid clue/board pairs. In order to gather more data on each clue/board pair, I randomly selected 400 of these pairs for use in study 2 and all subsequent analyses. Participants were again recruited from Prolific ($N = 201$, $M_{age} = 31.37$, $SD_{age} = 11.75$, 92 females, 7 other), this time to act as guessers. On each trial, participants were shown a clue followed by a board of words. They were told that the clue was chosen by their partner in order to to help them guess exactly 3 of the words on the board, and asked to select the 3 words that they thought their partner had in mind based on the clue. Each participant saw a total of 20 clue/board pairs randomly selected from the set of 400. They were shown the example board in Figure 3.2 before beginning the task.

After completing 20 trials, participants saw a randomly selected 7 of their clue/board pairs for a second time, and were asked to select the same words they had selected the first time they saw that clue/board pair. Trials which were included in this set of 7, and in which a participant selected more than 1 different word from their first selection were considered failed. Failed trials, and all the trials of participants who failed more than 3 trials, were excluded from subsequent analyses. 141 trials were excluded out of a total 4150, leaving an average of 11.02 valid responses for each of the 400 clue/board pairs.

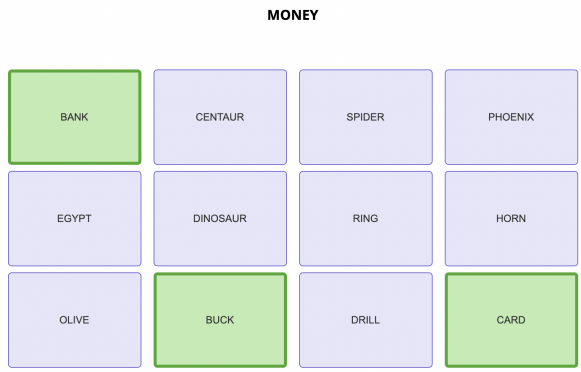

Figure 3.2: Each participant in study 2 saw this example before beginning the task, along with the following text: "In the example below, the clue money applies to the selected words bank, buck, and card. Money is kept in banks, a buck is a slang term for money, and people use cards to pay for things in place of money. While the other words may also bear some relation to the clue money, it seems likely these 3 selected words were intended by the clue."

Section 3.2

Results

Partnerships between spymasters and guessers were generally successful, with guessers guessing 79.21% of words correctly on average, compared to chance performance at 25%. Additionally, guessers got all 3 words correct on almost half of all trials (47.67%), with chance performance at just 0.45%. The standard deviation of percent correct responses over all the trials was 22.45%. This variability in success can be more attributed to spymasters than guessers; the standard deviation of percent correct responses given by each subject was 6.65%, while that of percent correct responses given to each clue/board pair was 14.88%. In fact, guesser responses to the same clue and board differed on average by less than one word. For each clue/board pair, any 2 subjects who saw that pair had on average a 76.31% percent overlap in their guesses, or 0.7106 different guesses out of the 3.

To generalize, guessers tended to agree on correct answers. This result supports

the idea that people draw on a common understanding of the subtleties of word relationships, and engage in similar cognitive processes when playing Codenames. It is thus theoretically possible to explain and emulate such a process computationally; the remainder of this paper will examine the extent to which current NLP approaches make progress towards this goal.

Chapter 4

Statistical Approaches

The NLP techniques explored here learn patterns present in large amounts of text. I first discuss bigrams, which reflect very simple patterns, and then move on to embedding-based approaches, which capture to varying degrees more of the complexity and nuance in human language. I compare the performance of each approach on the guessing portion of the Codenames task to human performance.

Section 4.1

Bigrams

A bigram is a set of 2 pieces of text that occur sequentially. Word bigrams in the previous sentence include (A, bigram), (bigram, is), (is, a), etc. Under the hypothesis that related words are more likely to co-occur, bigram frequencies might serve as a useful proxy for the relatedness of words. While this approach uses co-occurrence statistics in a less sophisticated way than embedding-based approaches, Shen et al. (2018) found that bigram frequencies outperformed both Word2Vec and GloVe embedding-based approaches in predicting human clues and guesses in a modified Codenames task [27]. Notably, players in this task chose clues from a set of adjectives, and guessed from a set of nouns.

4.1.1. Overview of Approach

I used freely available Google N-Grams to find the average probability of bigrams in a large corpus of english books from 2009-2019 [19]. The probability of a bigram is calculated by taking the proportion of all bigrams in the corpus that are the given bigram. To determine the similarity between two words, I took the larger bigram probability of the words in either order. For example, the similarity between the words "hot" and "dog" would be calculated as the larger of the bigram probabilities for "hot dog" or "dog hot." For each clue/board pair, I calculated in this manner the similarity between each board word with the clue, and selected the 3 most similar board words to be guessed.

4.1.2. Results

The bigram-based approach guessed 32.08% of words correctly, and guessed all 3 words correctly on 1.75% of trials, falling far below human performance. Its guesses overlapped with human guesses on just 31.35% of words on average. The difference in performance of this approach on this task, and that devised by Shen et al. (2018), can be explained by the fact that adjective-noun co-occurrence counts will be higher if the adjective is often used to describe the noun. Thus bigram probabilities are a good proxy for human judgements about adjective-noun relatedness. In this task, however, clues and board words are not limited to particular parts of speech, allowing for a greater variety in relationship types which are clearly not all well-captured by bigram probabilities.

In fact, only 15.5% of the clue/word pairs had a nonzero similarity score, or probability of bigram occurrence. Importantly, words that spymasters intended to be picked out by their clues were more likely than unintended words to have a nonzero similarity score; 20.66% of intended words had a nonzero score, while 13.78% of

unintended words had a nonzero score, and the average proportion of intended words with a nonzero score per board differed significantly from the average proportion of unintended words with a nonzero score per board ($t(400) = 5.804$, $p < 0.001$). This suggests that while bigram probabilities do capture a small portion of human word-relatedness judgements.

Section 4.2

Embedding-based Approaches

Each embedding-based approach discussed here maps words to vectors which represent their positions in a semantic space, where more semantically similar words are meant be positioned more closely in this space. To calculate the similarity between two words according to these embedding-based approaches, it is conventional to take the cosine similarity between the embeddings of those words. Here, I implement a naive guessing strategy for each embedding-based approach in which I select the 3 board words with the highest cosine similarity to the clue for each clue/board pair. Approaches and guessing strategies which incorporate more contextual information will be discussed in the next chapter.

4.2.1. Overview of Approaches

Word2Vec. Word2Vec vectors [20] are created using a neural network which is fed text and learns to predict words based on their context. A word's context is considered to be each word within a certain distance from it in the input text. In the text "Codenames is a fun game to play with friends," with a context window of 3 and target word "game," context words include "is," "a," "fun," "to," "play," and "with." The model I used, obtained from the gensim library [24], was trained on the Google News text corpus. It receives as input a target word, and outputs the probability of

the occurrence of possible context words. It is trained with negative sampling, where for each target word, the network computes the probability of just one of the context words and a set number of randomly sampled negative instances rather than that of each word in the vocabulary. During training, it learns to reduce the probabilities of the negative instances and increase that of the context words.

This particular model learns 300 dimensional vector representations of each word in the vocabulary. The vocabulary did not contain 8 of the clues given by spymasters in the selected clue/board pairs, and so 8 of the 400 clue/board pairs were excluded from analysis for this model.

Glove. Similar to Word2Vec, the GloVe (Global Vectors) model [22] is trained to predict the likelihood of words co-occurring. However, while Word2Vec learns from local contexts, with more frequently co-occurring words making up more of the training data, GloVe is explicitly trained on the corpus-wide co-occurrence statistics of words. The training objective is to learn word vectors such that words that are more likely to co-occur have more similar vectors. While variables such as the size and contents of the training corpus, word vector length, and number of training iterations have considerable effects on both models' performance, GloVe learns more efficiently than Word2Vec and tends to outperform it.

The GloVe vectors I used were pre-trained on Twitter and had 300 dimensions. The vocabulary did not contain 8 words from the clue/board pairs, and so I only evaluated the model on trials where the clue and each intended word was in the vocabulary. This condition excluded 8 out of the 400 clue/board pairs from analysis for this model. For 13 of the remaining 392 pairs, one unintended word on the board was not in the model's vocabulary. When calculating the model's guess for these clue/board pairs, I simply assigned a similarity score of 0 between this unknown word and the clue. This increased chance performance very slightly from 25% correct

guesses to 25.07% correct guesses.

Bert. Bert, or Bidirectional Encoder Representations from Transformers [5], uses a transformer architecture to take more context into account when predicting target words during training. Transformers are a deep learning approach in which every input is connected to every output, allowing the input to be processed in any order rather than strictly sequentially. The weights between inputs and outputs are dynamically calculated based on the input, a process known as attention. BERT has 12 layers of transformers, with the attention computation applied at each layer to bias the intermediate representations of words in a particular way based on the words around them. As part of the process in which it learns to map words to embeddings, it learns to what extent and in what manner it should take certain context words into account when computing the embedding of an input word.

The BERT model learns by masking a word in the input data, encoding all the words both proceeding and following this masked word within some window into a vector, and using this vector to predict the masked word. Technically, BERT actually learns representations of sub-words, splitting words into the most common sequences of characters. This allows BERT to produce vector representations for words it does not see during training. BERT is also trained to predict entire sentences; given a pair of sentences, it learns whether they are likely to be seen in succession. While BERT can produce vector representations of whole sentences or even paragraphs, in which each word's contribution to this vector depend on the words around it, it can also produce vector representations of context-free, single words. The performance of these context-free word embeddings is discussed below.

The BERT model I used was downloaded from the SentenceTransformer library [25]. It was trained on a diverse data set and produces vectors with 768 dimension.

GPT-3. GPT-3, released by OpenAI, is also a transformer-based model, but is far larger and trained on far more data than BERT [1]. GPT-3 is trained to repeatedly predict the next word in a sequence of text. In this way it is more simple than BERT, which is trained to predict a masked word based on the words that come before and after it, and also to predict pairs of successive sentence. GPT-3 has 175 billion parameters, or values which it learns to optimize during training. In contrast, BERT has over one thousand times less at 110 million parameters, and GPT-3's predecessor GPT-2 has 1.5 billion. GPT-3 has made surprising progress towards zero, one, and few-shot learning, and shocked users with its ability to generate text that seems as though it was written by a human. However, perhaps the most interesting aspect of GPT-3's success is its simplicity - while it has a very similar architecture to BERT and GPT-2, its massive increase in size and amount of training data led to noticeable differences and sensational public response [2, 8]. While GPT-3 is primarily used for text generation, OpenAI offers a variety of GPT-3 word embedding models, meant for different tasks and producing different sizes of embeddings. I used the four models that were specifically intended for determining semantic similarity of texts, and produced vectors with 1024, 2048, 4096, and 12288 dimensions, with the largest model being the most expensive and supposedly the most capable. I accessed these models from the OpenAI API.

4.2.2. Results

The performance of each model is detailed in Table 4.1. BERT embeddings performed best, guessing the highest proportion of words correctly, guessing all 3 words correctly for the highest proportion of clue/board pairs, and deviating the least from human guesses. It is interesting to note that GPT-3 embeddings performed worse than BERT and similarly to GloVe and Word2Vec embeddings, despite being much larger and being produced by a sophisticated and large model trained on far more data

Model Performance			
Model	Percentage of Correct Guesses	Percentage of Perfectly Guessed Groups	Percentage of Overlap with Human Guesses
Word2Vec	58.16	17.60	57.00
GloVe	58.59	15.30	57.43
BERT	64.25	23.00	62.63
GPT-3 (vector size 1024)	58.83	5.33	59.13
GPT-3 (vector size 2048)	52.50	3.66	52.53
GPT-3 (vector size 4096)	58.83	5.25	57.97
GPT-3 (vector size 12288)	53.08	3.83	53.43

Table 4.1: For each embedding-based model, the percentage of correct guesses given by this model, percentage of perfectly guessed groups (clue/board pairs in which all three words were guessed correctly), and percentage of overlap with human guesses, calculated by taking the percentage of guesses that the model had in common with each human guesser for each clue/board pair.

than the other models. Comparisons within the GPT-3 models are similarly hard to explain, with no clear relationship between model or word embedding size and performance. While larger models and embeddings can theoretically capture more information, much of the information captured by larger GPT-3 models is apparently not useful for the Codenames task.

There is a strong, positive correlation between each model's percentage of correct guesses and percentage of overlap with human guesses ($r = 0.9880$, $p < .001$). A positive relationship between these two variables is expected, as humans had a high percentage of correct guesses on average (79.21%). However, the strength of the relationship suggests that better performing models by and large capture more of what humans guessers are doing. If the words correctly guessed by each model were randomly distributed, the expected overlap between each model's guesses and human guesses would be expected to be on average 8.577% less than what is ob-

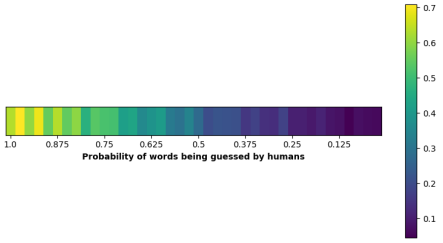

Figure 4.1: The probability of words being chosen by human guessers for a given clue/board pair, where color represents words' BERT embedding cosine similarity to the clue relative to other board words.

served. A dependent t-test between each model's actual overlap with human guesses, and expected overlap with human guesses if guesses were random, revealed that this difference is significant ($t(7) = -30.97$, $p < .001$).

While the BERT model had the most success out of the embedding-based models considered here, it still falls far below human performance. Humans on average guessed 79.21% of words correctly, and correctly guessed all 3 words on 47.67% of trials, with the BERT model at 64.25% and 23.00% respectively. Humans tended to agree with the BERT approach when it was correct; on average, humans guessed 88.28% of the words BERT guessed correctly. It is therefore unsurprising that BERT word embedding similarity also predicted the probability of words being chosen by human guessers. There is a strong positive correlation between the probability of each word being guessed by human guessers for a given clue/board pair, and the percentile of that word's similarity score to the clue, relative to other words on the board ($r = 0.9294$, $p < .001$). See Figure 4.1 for a visual depiction of this relationship.

Chapter 5

Incorporating Context

In the approaches described so far, static representations of words have been compared in isolation to determine word similarity. However, in Codenames, both spymasters and guessers may iteratively reason about possible word meanings and determine how well clues apply to board words in the wider context of all the relevant words. Several approaches for emulating these dynamic, contextualized judgements are described in this chapter. First I discuss guessing strategies in which each of the 3 words in a potential group of guessed board words is taken into account when determining the quality of a guess, rather than considering each word individually. I then consider approaches for determining word similarity in which more contextual knowledge is included in a word's embedding. Finally, I attempt to determine similarity by comparing contextualized representations of words, which favor the most relevant senses of polysemous words.

Section 5.1

Guessed-Word Group Relationships

When coming up with a clue for a given board, one might generate sets of highly related words to a single board word at a time before deciding on a clue based on

the intersections of these sets. However, this process likely requires the generation and evaluation of many bad clues. Spymasters might instead employ heuristics based on similarities between the limited set of board words. For example, if two board words are obviously related (e.g. "turkey" and "robin"), a spymaster might consider clues that capture this relation ("bird," "animal," "wings," etc.), and then tweak or expand this clue set to include a third board word.

While spymasters may not go through this exact mental process, the average BERT embedding similarity between pairs of intended words on each board indeed differs significantly from the average BERT embedding similarity between pairs of intended and unintended words on each board ($t(400) = 19.18$, $p < .001$). Pairs of intended words have an average similarity of 0.2995 while pairs of intended and unintended words have an average similarity of 0.2300. This suggests that more similar groups of words are more likely to be intended by a clue. We can calculate the inter-group similarity of a group of words by taking the average similarity between each pair of words in the group. Then, we might rank each group of 3 board words which might be intended by the clue according to their inter-group similarity, where a higher rank indicates greater inter-group similarity. The number of words guessed correctly by the BERT embedding based approach per clue/board pair correlated positively with the guessed group's inter-group similarity ranking ($r = 0.2121$, $p < .001$), which supports the intuition that guessers are more successful when the words they guess are more similar to each other, relative to other groups of words they could have guessed.

5.1.1. Approach 1

In the initial guessing strategy for the BERT embedding based approach, we simply select the three most similar words to the clue, which is equivalent to choosing the word group with the greatest average similarity to the clue. To incorporate inter-

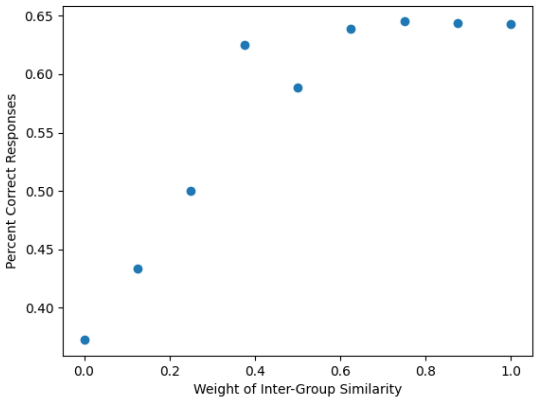

Figure 5.1: Percentage of words guessed correctly using Approach 1, where the word group chosen for each board is that which has the highest weighted average of inter-group similarity and average similarity to the clue. The weighted average is calculated as $w1*$(inter-group similarity)$+(1-w1)*$(average similarity to clue) where $w1 <= 1$. Performance is reported at various values of $w1$, where higher values mean that inter-group similarity is taken more into account relative to average similarity to the clue when determining what group of words to guess.

group similarity into the guessing strategy, we might calculate a weighted average between inter-group similarity and average similarity to the clue for each group. The performance of this guessing strategy with various weights is detailed in Figure 5.1. With this guessing strategy, performance is not meaningfully improved by taking inter-group similarity into account.

5.1.2. Approach 2

Instead of incorporating the similarity between guessed words into a static score for each word-group, we might use the same principles to iteratively build a word-group in a less computationally expensive manner which better emulates the proposed heuristics for how humans play Codenames. In such a process, board words might be iteratively guessed based on their similarity to some context. The clue serves as an initial context, and the most similar board word to the clue is added to the word-

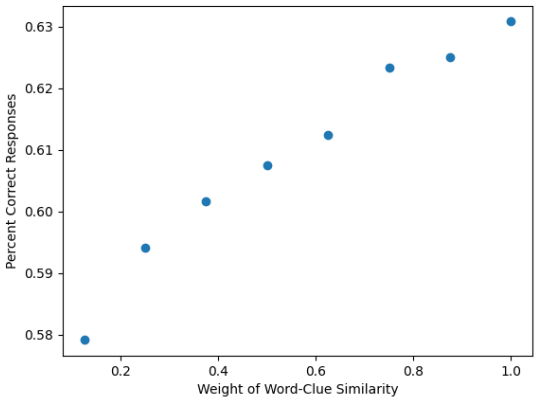

Figure 5.2: Percentage of words guessed correctly using Approach 2, where guessed words are chosen iteratively based on the weighted average of their similarity to the clue and other context words, calculated as $w*$(similarity to clue)$+(1-w)*$(average similarity to other context words). Performance is reported at various values of w.

group and also to the context. The next word is then chosen based on its similarity to the context, which now consists of the clue and the previously chosen word. This process repeats once more, and a third word is added to the word-group based on its similarity to the context consisting of the clue and previously chosen words.

The similarity between a word and a multi-word context might be determined in various ways. I first took the weighted average between the word and each word in the context, where the similarity of the word with the clue had some weight w, and the average similarity of the word with the remaining words in the context had weight $1-w$. The performance of this guessing strategy at various values of w is detailed in Figure 5.2. Similarly to Approach 1, taking a word's similarity with previously chosen words into account did not improve performance.

5.1.3. Approach 3

Using the same iterative process as Approach 2, the similarity between a word and a multi-word context might instead be determined using BERT sentence embeddings.

Approach 2 considered the similarity between a word and each context word separately. By comparing a word to an embedding of the entire context, we might consider how likely these words are to co-occur all together, so that words which are linked to the clue in a similar manner will be more likely to be guessed. A context embedding can be constructed by combining the context words into a phrase, such as by separating each word by "and." The BERT model, trained to represent groups of sub-word tokens rather than single words, can then encode this entire phrase into an embedding, which will be similar to the embeddings of words or phrases with which the context is likely to co-occur. This approach guessed on average 64.17% of words correctly, performing slightly worse than the original BERT embedding based approach in which only a word's similarity to the clue was considered.

Across the three approaches for incorporating word-group similarity into the guessing strategy, no approach meaningfully improved the performance of the original BERT embedding based approach. While I do not report specific data on how well each approach emulated human guesses, none of the approaches, under any of the tested conditions, improved on the original approach. Although correctly guessed words are more likely to have similar BERT embeddings to each other, this pattern seems to have already been implicitly captured by the original guessing strategy. Words which have similar embeddings to that of the clue must occur in similar contexts to the clue, and therefore each of the most similar words to the clue likely occur in similar contexts to each other, causing them to have similar embeddings. Thus, it is understandable why biasing guesses towards words that are more similar to each other does not improve the performance of the original approach, even if this process does indeed capture a heuristic that humans employ.

Section 5.2

Contextualized BERT Embeddings

Another shortcoming of the current embedding-based approaches is the fact that many words have multiple meanings, and cannot be well-represented by a single point in semantic space. For example, in a vocabulary consisting of the words "river," "bank," and "money," we would want the words "river" and "bank" and "bank" and "money" to occupy similar regions of semantic space, but "river" and "money" to occupy very different regions of semantic space. These are opposing goals, and cannot both be prioritized. Thus, while BERT word embeddings are inherently limited in how well they can capture a word's multiple meanings, humans may flexibly assign any number of meanings to a single word (even contradictory ones - these are "contronyms," such as the verb "dust").

While embeddings for single words are forced to combine a word's different meanings into one representation, it is possible to bias the representation towards certain meanings by taking context into account when creating the embedding. The BERT model is trained to learn representations for sequences of sub-word tokens, based not only on the tokens but also on the way they are combined in the sequence. When encoding a phrase, the model will encode each token in that phrase while taking that token's position and the context tokens into account, and then combine these token encodings to obtain the embedding for the entire phrase [5]. So, in the process of encoding the phrase "river bank," the model will create an embedding for the word "bank" that incorporates the fact that it is in the context of the word "river."

In Codenames, clues may be given such that one particular meaning of the clue relates to one particular meaning of an intended word, even if both words have many other possible meanings that are unrelated. Thus, when comparing board words

to clues, guessers must consider both words in the context of the other to evaluate relatedness. I attempted to emulate this process using BERT sentence embeddings. To calculate the similarity between a clue and board word, I combined the two words into a phrase, encoded this phrase with the BERT model, and extracted the token embeddings for each word. I hoped that these contextualized word embeddings would represent the senses of each word that are most similar to the other word, emulating the human ability to think flexibly about multiple word meanings when comparing two words.

While I tried many different versions of this approach, organizing the two words in various ways in the phrase, and comparing all combinations of contextualized and non-contextualized clue and board word embeddings, all of these versions guessed between 45% and 50% of words correctly, falling far below the performance of the original BERT embedding-based approach. While I attempted to control for the position information encoded in the token embeddings by averaging the embeddings of a word at various positions in a phrase, and comparing words which occured in the same position in a phrase, it is possible that the position information encoded by the model corrupted the purely semantic information that is captured by single-word embeddings.

Additionally, contextualizing the word embeddings in this way tended to inflate similarity scores overall and decrease the variance between similarity scores. Rather than usefully representing a particular meaning of a word based on the context, token embedding are more like intermediates between the uncontextualized word embedding and its context. The inflation in similarity scores between clues and board words is not uniform; because of BERT's attention mechanism, which determines how word tokens affect the encoding of the other words tokens in the input, the extent to which token embeddings of board words are biased towards the clue word in the context

depends on what those board and clue words are. For example, consider the token embeddings of the word "bank" extracted from the inputs "river bank," "the bank," and "bank." The uncontextualized word embedding for "bank" is more similar to the token embedding for "bank" from the input "the bank" than from the input "river bank," indicating that "river" is attended to more than "the" when encoding "bank" in the two inputs. Importantly, the embedding for "bank" is more similar to that for "river" than "the," and so this result would not have occurred if the model simply biased the embedding towards its context in a uniform manner.

The above result is intuitive, since "river" modifies the meaning of the uncontextualized word "bank" more than "the" does. However, if we consider the token embedding for "bank" in the context of something unrelated, such as in "table bank" or "flavorless bank," attention comes into play to an even greater extent, so that the uncontextualized word embedding for "bank" is more different from "bank" in "table bank" or "flavorless bank" than it is from "bank" in "river bank." This indicates that, although embeddings are not biased uniformly towards context words, they may tend to be biased more towards unexpected or unrelated contexts than expected contexts. Indeed, when considering the contextualized board word embeddings in the guessing task (token embeddings extracted from a phrase of the form "'clue' and 'board word'"), the embeddings for board words that the spymaster intended to be guessed were biased less than those of unintended words. For example, on one clue/board pair with the clue "restaurant," intended board word "data," and unintended board word "robot," the embedding of "date" in "restaurant and date" was more similar to the unonctextualized embedding for "date" than the embedding for "robot" in "restaurant and robot" was to the uncontextualized embedding for "robot." This pattern remained on a large scale, with the average similarity between contextualized and uncontextualized intended words being 0.6588, while the average similarity

between contextualized and uncontextualized unintended words was 0.6447. A dependent t-test revealed that the difference between the two groups was significant across clue/board pairs ($t(400) = 4.773$,$p < .001$).

While I had hoped that the contextualized word embeddings would represent a particular word sense when the context word related to that word sense, and would be similar to uncontextualized word embeddings when the context word was unrelated, this was not the case. Instead it appears that when calculating token embeddings during training, the disambiguation of word senses was not as useful as attending to unrelated or unexpected context words. Since the model is trained to use sentence embeddings to predict masked words and next sentences, perhaps the disambiguation of word senses is not particularly useful, because the applicable word sense depends on the context which is already being incorporated into the sentence embedding.

Section 5.3

BERT Definition Embeddings

Given the importance of encoding entire sentences during training of the BERT model, I next considered whether BERT embeddings of dictionary definitions of words might produce richer, more human-like representations of those words. I took dictionary definitions of each clue and board word provided by the python NLTK WordNet package[17, 7]. I then used the BERT sentence embeddings of these definitions as the embedding for words when calculating the similarity between clues and board words. This approach was not very successful, guessing only 34.55% of words correctly and all three words correctly on 1.302% of clue/board pairs. Since dictionary definitions explicitly state the important aspects of a word's meaning, the poor performance of this approach indicates that BERT sentence embeddings do not encode this text in a human-like way.

Section 5.4

GPT-3 text prediction

While GPT-3 word embedding distance was a surprisingly poor proxy for human judgements in the Codenames task considering the size of the model and amount of training data, this word similarity metric may not capture the strengths of the model. GPT-3 is trained to predict the next word in a sequence, and because of its size can encode and reproduce many nuanced patterns present in its (499B tokens of) training data [2, 1]. Similarity metrics that take better advantage of GPT-3's ability to generate text in a human-like way are likely to have more success on the task.

5.4.1. Approach 1

One way to extract human-like, conceptual knowledge about words from GPT-3 is to simply prompt it to write a paragraph about that word. For example, when prompted to "write a paragraph about the word 'platypus'", GPT-3 responds: "The word 'platypus' is derived from the Greek word 'platypodes,' meaning 'flat-footed.' The platypus is a small, semi-aquatic mammal found in eastern Australia. It is the only known mammal that lays eggs. The platypus is known for its bill, which is similar to that of a duck. It also has a furry body and webbed feet." We can then ask GPT-3 to produce an embedding for this paragraph, and compute the similarity between words through the cosine similarity of their paragraph embeddings. With this approach, only 47% of words are guessed correctly with the largest model, which is worse than the performance of the GPT-3 word embedding based approach at every size of the model. Although the production of paragraphs from words forces the model to explicate conceptual knowledge about each word, some randomness is introduced in what specific topics the model happens to emphasize, which may account for the drop in performance.

5.4.2. Approach 2

We might avoid whatever particular limitations GPT-3 embeddings have and take better advantage of GPT-3's ability to generate text in a human-like way by simply asking it to do the Codenames guessing task in the same way humans were asked to do it. For each clue/board pair, I asked GPT-3 variations of the question, "Which three of these words are most related to 'clue'?", followed by the list of board words. On some trials GPT-3 listed more than 3 words, or words not included in the list, in which case I took the first three board words listed. With this approach, GPT-3 guessed 61.75% of words correctly, guessed all three words correctly on 22.0% of trials, and on average chose 59.04% of words in common with human guessers. Relative to other approaches, GPT-3 is successful at guessing words correctly, although its performance still falls below that of the original BERT embedding based approach. While GPT-3 had almost as many perfectly guessed trials as the original BERT embedding based approach (22% and 23% of trials perfectly guessed, respectively), its variance in number of correctly guessed words per clue/board was higher, at 0.7357 as compared to 0.5672 for the BERT embedding based approach. This suggests that the random elements in the GPT-3 responses lead the approach far astray in certain cases, whereas other models may be more likely to be just slightly off more often. There is a fairly significant overlap in the words that the original BERT embedding approach gets correct with this approach, with the BERT approach guessing 75.61% of the words that this approach guessed correctly. This overlap is not surprising, since the models are similarly trained to predict words based on preceding or surrounding text.

While this approach performs similarly to the original BERT embedding-based approach, it has the disadvantage of being even less transparent. Some responses seemed extremely difficult to justify. For example, for the clue "outside," with intended words

"well," "yard," and "deck," this approach instead guessed the words "straw," "post," and "yard." While guesses varied slightly with rewordings of the prompt (none of which improved overall performance), "straw" was consistently guessed. When I prompted GPT-3 to justify its answers, it explained that "a straw is something you drink outside," which is not a response I would expect to hear from a human, and indeed may have nothing to do with why GPT-3 actually gave that response. Because GPT-3 simply predicts next words given past words, the explanation need not have actually justified the response - rather GPT-3 deemed that explanation likely to occur in text given that it is preceded by such a question and response. While "straw" and "outside" clearly have some relation in the text GPT-3 is trained on, it is not at all clear what that relation is.

Chapter 6

Knowledge-Based Approaches

The statistical approaches described in the previous section all rely on patterns in text to learn representations of words. Words which tend to occur in similar contexts or tend to co-occur play similar roles during training, and so these models learn similar representations for them. With enough training data and a large enough model, there is indeed much information that might be learned from statistical patterns in text; however, these patterns might not always accurately reflect human knowledge about how words relate to each other.

Language is used to symbolically communicate thoughts, and so text, as written language, reflects these thoughts. However, there are many mismatches between the conceptual space accessed by human minds and the patterns present in text. For example, the frequency with which certain types of things occur in certain real-world contexts or have certain attributes often do not match the frequency with which they are described in these contexts or in terms of these attributes in text. By way of illustration, consider a banana. I would guess that when picturing a banana, most people picture something yellow. While bananas can be other colors, such as green, yellow-ness seems to be a characteristic attribute of bananas. In text, however, people are more likely to talk about "green bananas" than "yellow

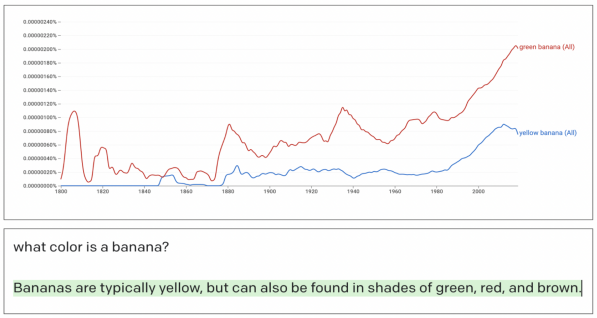

Figure 6.1: Above, google N-gram probabilities for "green banana" and "yellow banana." Below, a discussion with GPT-3 about bananas.

bananas," precisely because bananas are generally considered to be yellow, making it useful to specify when they are green (Figure 6.1). Language models such as BERT or GPT-3 are trained to learn more than just bigram probabilities, and thus can learn to associate bananas with yellow-ness from other patterns in text. For example, in text where people talk specifically about the colors of things, or are describing bananas in detail such as in a dictionary definition, the idea of bananas being yellow is probably more likely to come up than the idea of bananas being green. So, when asked what color bananas are, GPT-3 recalls similar training text and outputs a variation of what tended to follow this text - which reflects human beliefs (Figure 5.1). Interestingly, GPT-3 word embedding cosine similarities do not reflect the same ideas as this response, with "brown" and "banana" having a cosine similarity of 0.8316, "yellow" and "banana" at 0.8224, "green" and "banana" at 0.8043, and "red" and "banana" at 0.771. Although pink bananas were not mentioned in GPT-3's response, "pink" is determined to be more similar than "red" to "banana," at 0.7952.

This is by no means a rigorous test of what information large scale language models might capture from text, but rather an illustration of the mismatch between statistical patterns in text and in human thought. Language models must learn something other than these patterns in order to have human-like representations of words.

Knowledge-based approaches explicitly state relationships between words that humans consider to be important. In the knowledge graphs described below, words are represented as nodes, with edges between related words, and edges labelled by relationship types. Constructing these graphs is more laborious than the embedding-based approaches previously discussed, as they often require human decision-making about what words are related and how.

Section 6.1

Overview of Approaches

6.1.1. ConceptNet

ConceptNet [28] has been continuously constructed since 1999, when it was first launched as a crowdsourcing project in which people were recruited to simply state relationships between object or events, with the goal of explicitly stating the most basic things that people know. These statements were then parsed into the graph structure described above. More recently, ConceptNet has grown to include knowledge from other crowdsourcing initiatives as well as extracted knowledge from Wikipedia and online dictionaries. Edges in ConceptNet are labelled by the relationship type between the words (Antonym, DistinctFrom, EtymologicallyRelatedTo, LocatedNear, RelatedTo, SimilarTo, Synonym, AtLocation, CapableOf, Causes, CausesDesire, CreatedBy, DefinedAs, DerivedFrom, Desires, Entails, ExternalURL, FormOf, HasA, HasContext, HasFirstSubevent, HasLastSubevent, HasPrerequisite, HasProperty, InstanceOf, IsA, MadeOf, MannerOf,MotivatedByGoal, ObstructedBy, Part), and also by a weights which denotes how reliable an edge is based on its source. Words may be connected by more than one edge. For example, most words connected by any edge will also be connected by a "RelatedTo" edge. Given the scope of the project, it is no surprise that ConceptNet is not perfect; the creator herself stated in a chat

forum that "most objects will be missing most properties that apply to them."

While there are various methods for calculating semantic similarity of words in a knowledge graph ([30, 15, 13, 10, 21], path length is often key to these calculations. However, due to the huge variety of edges in ConceptNet, words tend to be highly connected. In fact, for each clue/board pair, clues that were included in ConceptNet had a maximum path length of 2 from each board word, making path length a poor similarity metric for this task. I instead considered whether or not words were directly related to each other, as well as the number of neighbors they had in common.

Three of the clues given by spymasters were not included in ConceptNet, and so I did not include the corresponding 3 clue/board pairs in my evaluation of ConceptNet.

6.1.2. WordNet

WordNet [7] is similar to ConceptNet, but has a few structural differences. While the nodes in ConceptNet are words, those in WordNet contain synsets, or groups of words that might be used interchangeably in a certain context. Additionally, most of the related words in WordNet are the same part of speech, so that it can be divided into sub-nets containing nouns, verbs, adjectives, and adverbs, with the few links across sub-nets often being morphosemantic relationships, such as that between "jog" and "jogger." WordNet also captures mainly hierarchical ("is a", "part of") relationships between words, while ConceptNet captures more diverse relationships.

The structure of WordNet makes path length a more appropriate similarity metric. Because nodes are synsets rather than words, polysemous words appear more than once in the graph and are surrounded by neighbors which relate only to that synset's word sense. This prevents the occurrence of very short path lengths between words with very different meanings when those words are both highly related to different meanings of another word. I determined the similarity between two words by finding all synsets containing one of the words, and then finding the shortest path length

(using the NLTK python package [17]) between any of these synsets and any synsets containing the other word.

Eleven of the clues given by spymasters were not included in WordNet, and so I did not evaluate WordNet on these clues.

Section 6.2

Results

The WordNet approach correctly guessed only 34.55% of words, and guessed all 3 words correctly for 1.302% of clue/board pairs, falling far below the performance of the vector-based approaches but still exceeding chance performance at 25% and 0.45% respectively. Accordingly, the approach differed more than embedding-based approaches from human guessers, guessing on average 34.02% of words in common with each human guesser. The poor performance of this approach might be explained by the strict grouping of parts of speech, or by the fact that words in a synset might have subtly different meanings which limit the relationships that can accurately be drawn to other synsets. Navigating WordNet in a different way might yield better results, however I do not explore alternate approaches here. Instead I consider ConceptNet, whose overarching goal of stating as many relations between words as possible seems to be a better fit for the task.

While the ConceptNet approach is indeed more successful than WordNet, it still falls below the performance of almost all of the embedding-based approaches. When calculating word relatedness based on the number of shared neighbors between words, the approach correctly guessed 50.83% of words, and guessed all 3 words correctly for 10.00% of clue/board pairs. When I instead summed the edge weights connecting both words to their common neighbors, performance increased slightly to 5183% and 10.75% respectively. Using this same approach, but giving the highest related-

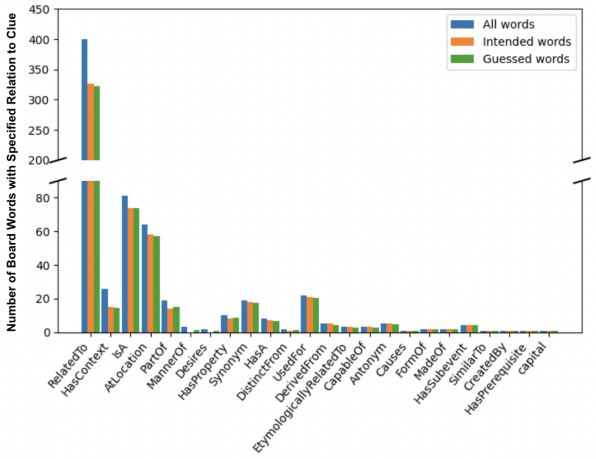

Figure 6.2: Across clue/board pairs, number of board words with each type of relation to the clue, as specified in ConceptNet. Reported for all words, words the spymaster intended to be guessed, and words guessed by human guessers (averaged across participants). Words can have more than one type of relation to the clue, in which case all are reported.

ness score to words which are directly related, performance increased to 54.17% and 13.50%. While it is likely that other methods might further boost performance, I shifted my analysis to more precisely characterizing what aspects of human word-relatedness judgements ConceptNet captures.

Surprisingly, only 1.205 words per board on average were directly related to the clue word in ConceptNet. Word relationships in ConceptNet thus failed to capture the majority of relationships that humans identified between clues and intended or guessed words. However, the relationships that ConceptNet did identify accurately reflected human judgements. When board words had a ConceptNet relationship to the clue, spymasters intended these words to be guessed 79.46% of the time, and guessers guessed these words 80.53% of the time. There was no particular ConceptNet relationship type that humans seemed to favor over others; for a breakdown of the relationship types between clues and all board words, intended words, and guessed words, see Figure 6.2.

Chapter 7

Combined Approaches

While knowledge graphs such as ConceptNet and WordNet flexibly contain information that humans explicitly deem to be true and important, they are laborious to construct and incomplete. Conversely, embedding-based approaches are complete, but are limited to representing patterns in text which may be inconsistent with human knowledge, and struggle to represent a word's multiple meanings. Given the complimentary nature of these approaches, I explored several methods for combining their strengths.

Section 7.1

Incorporating ConceptNet Relations

While ConceptNet might in principle capture relationships between words that would be hard for the embedding-based models to learn, in practice most of these relationships were already captured by the best embedding-based approaches. Only 1.205 words per board on average had a ConceptNet relationship to the clue, and the BERT embedding approach alone guessed 85.6% of these words already. The BERT embedding approach also correctly guessed 76.54% of the words correctly guessed by the best version of the approach which considered shared neighbors in ConceptNet.

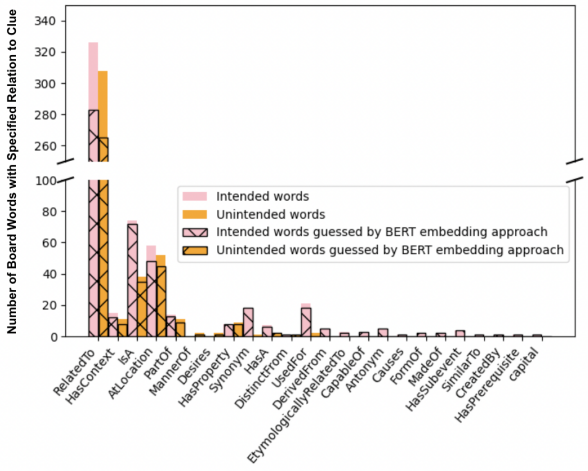

Figure 7.1: Across clue/board pairs, number of board words with each type of relation to the clue, as specified in ConceptNet. Reported for words the spymaster intended to be guessed, and which of these intended words the BERT embedding-based approach guessed, as well as words the spymaster did not intend to be guessed, and which of these unintended words the BERT embedding-based approach guessed. Words can have more than one type of relation to the clue, in which case all are reported.

While there is some difference between what ConceptNet and BERT embeddings capture, this difference is difficult to characterize. No ConceptNet relationship are particularly over or under-emphasized by the BERT embedding based approach, as can be seen in Figure 7.1.

I modified the BERT embedding-based approach by giving the highest similarity score to board words that were directly related to the clue in ConceptNet. While this did slightly improve performance on the guessing task, from 64.24% to 65.08% of words guessed correctly, unsurprisingly this improvement was not significant across clue/board pairs ($t(400) = -1.132$, $p = .2580$).

Section 7.2

Finetuning Embeddings with ConceptNet

7.2.1. ConceptNet NumberBatch Embeddings

In their description of ConceptNet, Speer et al. [28] recognize the benefits of combining the knowledge contained in ConceptNet with pre-trained word embeddings, which are produced using data that might be inaccessible or labor intensive to incorporate into the ConceptNet graph structure. They introduce ConceptNet NumberBatch embeddings, which are created by retrofitting GloVe and Word2Vec word embeddings on ConceptNet. The retrofitting process is described by Faruqui et al. [6] and involves iteratively biasing each word embedding to be closer to the embeddings of some group of other words, in this case its neighbors in ConceptNet. This forces the word embeddings to better express the relationships between words that ConceptNet captures. The retrofitted GloVe and Word2Vec embeddings were concatenating, and then reduced in dimensionality to 300, to obtain one word embedding for each word in the vocabulary. Since the retrofitting process tends to bias embeddings towards those for highly connected terms, the authors then subtracted the mean embedding from all retrofitted embeddings. These trained NumberBatch embeddings outperformed other models such as Word2Vec on word relatedness tasks.

Using pretrained NumberBatch embeddings for the Codenames task, guesses for each clue/board pair were calculated in the same way as the other vector-based approaches, by taking the three board words whose embeddings had the highest cosine similarity to the clue word embedding. Three of the clues given by spymasters were out of the vocabulary of the NumberBatch embeddings, and so these 3 clue/board pairs were excluded from analysis. This approach performed well relative to other approaches on the Codenames task, guessing 66.25% of words correctly on average,

guessing all 3 words correctly for 24.43% of clue/board pairs, guessing on average 64.54% of words in common with human guessers on each clue/board pair.

While these statistics indicate a slight boost in performance over the BERT embedding based approach, the differences in performance on both guessing words correctly and overlap with human guesses were not significant (for a dependent t-test between the 2 approach's performance on each of the 400 boards, $p = .2963$ and $p = .2279$ respectively). There was a significant, positive correlation between the 2 approaches' similarity scores for clues and board words ($r = .7449$, $p < .001$). Accordingly, the BERT approach guessed 78.89% of the words that the NumberBatch approach guessed correctly, while the NumberBatch approach guessed 80.83% of the words that the BERT approach guessed correctly, suggesting that there is substantial (though not complete) overlap between the approaches. Considering only the words that the NumberBatch approach guessed correctly, and which the BERT approach did not guess, either the original Word2Vec or the original GloVe embedding-based approach guessed 50.82% of these words correctly, suggesting that a portion of the advantage that the the NumberBatch approach has on the BERT approach may be due to differences between BERT embeddings and Word2Vec or GloVe embeddings, rather than the retrofitting process.

Chapter 8

Multi-Modal Embeddings

Of the approaches discussed, unsupervised approaches which learn word relationships through statistical patterns in large amounts of data, such as BERT embeddings and GPT-3 text prediction, have outperformed knowledge-based approaches in which word relationships are explicitly described by humans, and seem to account for much of what these knowledge-based approaches capture. This shows that these embedding-based models can effectively learn about abstract word relationships from statistical patterns in text. It is plausible that larger models trained on more text and perhaps in a more sophisticated manner might do so even more effectively. However, the fact remains that these models are still inherently limited in that they cannot learn more than what is represented in the text they are trained on, and other types of knowledge may comprise a portion of the gap between human and artificial language understanding.

To address this limitation, work has been done on multi-modal language models, which learn word embeddings from text as well as other sources such as images [18, 12, 4] or audio [29] (Liang et al. provide a recent review [16]). This work describes approaches for combining multi-modal data in various ways and at various stages of training. Here, I explore just one of many possible multi-modal embedding models.

Section 8.1

Approach

I used the pretrained OpenAI Clip Model, which is trained to predict which caption goes with which image, and produces 768-dimensional embeddings for both images and text [23]. The model thus learns representations of words that reflect visual depictions of those words. By guessing the 3 words with the highest embedding cosine similarity to the clue for each clue/board pair, this model guesses 43.74% of words correctly, and guesses all 3 words correctly for 5.542% of clue/board pairs. This is a clear drop in performance from the purely text-based statistical approaches. Of the words that the Clip model guessed correctly, the GPT-3 text prediction approach correctly guessed 75.05%, the BERT embedding based approach correctly guessed 76.01%, and the NumberBatch embedding based approach correctly guessed 79.08%, suggesting that these approaches already take into account many of the word relationships captured by these multi-modal embeddings that are important for the Codenames task. While BERT embedding similarity scores and NumberBatch embedding similarity scores correlated positively with the Clip embedding similarity scores ($r = 0.3428$, $p < .001$, and $r = 0.3400$, $p < .001$ respectively), this correlation was far lower than that between the BERT and NumberBatch approaches ($r = .7449$, $p < .001$). While the Clip embeddings do capture information that is different from what the purely text-based embeddings capture, this extra information does not seem to be useful for the Codenames task.

Chapter 9

Limitations of Current Approaches

A variety of approaches for emulating human performance on the Codenames guessing task have been evaluated and compared. All of these approaches fell short of human performance, with the most successful reaching about 80% of the average human guesser's success rate. Here I more specifically characterize the practical and theoretical limitations of these approaches in relation to human performance.

Section 9.1

Relationship Types Captured

The BERT embedding-based approach captured much of the useful information that could be extracted from knowledge-based approaches, multi-modal embeddings, and contextual information. Given this general trend, I hoped to better characterize exactly what aspects of human knowledge this approach was missing. While comparisons between the BERT embeddings and ConceptNet did not yield any clear findings as to particular relationship types that BERT captured more or less than others, a clearer pattern might emerge if humans are asked to describe how words are related rather than relying on ConceptNet labels. To this end, I identified clue/board word pairs where BERT similarity judgements differed from human judgements. I iden-

tified 100 clue/word pairs from each of four groups described in Table 9.1. I then recruited participants from Prolific ($N = 99$, $M_{age} = 31.32$, $SD_{age} = 12.25$, 40 females, 1 other) to decide whether words were related, and if so how. Each participant saw 32 clue/board pairs randomly sampled from each of the four groups, and for each pair was asked to rate on a scale of 1-100 how related they considered those words, and to describe how they were related (or to write "I don't think these words are related").

Clue/Word Pair types				
Pair Type	Board Word Intended?	BERT Guessed?	Humans Guesser Behavior	Example clue, word
False Positive	No	Yes	Least often Guessed	opera, beach
False Negatives	Yes	No	Most often Guessed	alive, duck
Easy True Positives	Yes	Yes	Most Often Guessed	eggs, ham
Hard True Positives	Yes	Yes	Least Often Guessed	song, fire

Table 9.1: Four groups of clue/board word pair types, chosen to highlight what BERT over and under emphasizes relative to humans. Human guesser behavior is reported within the category of clue/board word pairs described by the Board Word Intended? and BERT Guessed? columns.

While I had hoped to compare human explanations of relatedness from these different groups to better understand what relationship types BERT over and under emphasizes relative to human guessers, these explanations were extremely difficult to group systematically. While some explanations included clear relationships such as "isA" or "partOf", the majority were much harder to categorize, such as "Webcams are almost always linked to servers in some way," "White rabbits are referenced a lot," "A knife chops ingredients usually," "Packages come in the mail," "I think of them together," and "similar vibe," as well as many others. As a future direction of this project, I might recruit additional participants to group explanations which express

similar relationships. The difficulty of systematically grouping just this small set of human explanations of relatedness demonstrates an important challenge for NLP approaches such as ConceptNet which aim to explicitly represent word relations.

As for numeric human relatedness judgements, these varied as expected by category, with average relatedness ratings of 21.19 for false positives, 73.05 for false negatives, 84.21 for easy true positives, and 50.42 for hard true positives. BERT embedding similarity scores were only slightly correlated with human relatedness judgements ($r = 0.2384$, $p < .001$), however, this might be explained by the fact that most clue/word pairs were specifically chosen to highlight differences in human and BERT guessing behavior. Interestingly, though, even within the Easy True Positives category, in which BERT and most humans guessed the intended word correctly, the relationship between relatedness scores was similar ($r = 0.2691$, $p < .001$).

Section 9.2

Isolated Judgements

Most of the guessing approaches I tried considered relatedness between clue and board words in isolation, before taking the 3 most highly related words. While I explored ways of incorporating the wider context of a Codenames board in Chapter 5, none of these approaches meaningfully outperformed the original BERT embedding based approach which made isolated relatedness judgements. However, given that this sort of approach is possibly quite different from how humans likely play Codenames, by iteratively reasoning about how and to what extent board words are related to a clue as compared to each other, I wondered whether the isolated nature of the BERT embedding judgements indeed imposed a ceiling on performance.

I gathered data on isolated human judgements of word relatedness in order to use these isolated judgements in a naive guessing approach that mirrored the isolated

judgements of the BERT embedding based approach. With such an approach, the role of human relatedness judgements might be separated from humans' ability to flexibly account for contextual information in human success on the task. I randomly selected 34 clue/board pairs from the subset of 400 on which human and BERT success rates were not significantly different from the those for all 400 boards. For this subset of clue/board pairs, humans guessed 77.64% of words correctly, as compared to 79.21% on all 400 boards ($t(33) = -0.8660$, $p = .3927$). The BERT approach guessed 63.81% of words correctly for the subset of 34 boards, as compared to 64.25% on all 400 boards ($t(33) = -0.4005$, $p = .6914$). For this approach, relatedness judgements must be made between the clue and each of the 12 board words for each clue/board pair. I recruited participants from Prolific ($N = 59$, $M_{age} = 35.69$, $SD_{age} = 13.39$, 35 females, 2 other) to make these judgements. I asked each participant to rate on a scale of 1-100 the relatedness of 34 clue/board word pairs. Each participant rated one clue/board word pair from each clue/board pair, so no one saw the same clue twice.

With this data, I evaluated the performance of an isolated human relatedness judgement based approach on the Codenames guessing task. To compute a guess for each clue/board pair, I simply chose the three board words with the highest average relatedness ratings to the clue. This approach outperformed the BERT approach, guessing 82.35% of words correctly. Interestingly, it also beat the average performance of human guessers on the 34 boards ($t(34) = 2.447$, $p = .0199$). This approach predicted human guesses well; the guesses produced by this approach overlapped with human guesses by 77.36% on average, while human guesses overlapped with each other by 76.31% on average. These results indicate that isolated human relatedness judgements from different humans perform just as well on this task as contextualized relatedness judgements by single participants (both averaged across participants).

The fact that no two relatedness judgements for a clue/board pair came from the same participant, and that judgements from different groups of participants could be effectively compared on the same scale, is consistent with prior work which has found that people tend to agree on semantic similarity for both similar and relatively dissimilar words [3]. Most importantly, this shows that isolated relatedness judgements can in principle achieve the performance of human guessers on this task. The difference in performance between human and artificial guessers such as BERT must then be attributed to differences in these relatedness judgements.

I compared human relatedness judgements with relatedness judgements by embedding-based approaches on these 34 boards. I focused on the BERT embedding based approach and the numberBatch embedding based approach, which performed similarly and outperformed the other approaches. I also focused on two of the GPT-3 embedding based approaches: the largest GPT-3 embeddings, and also the smallest (which performed best out of the GPT-3 embeddings). All approaches were positively correlated with human relatedness judgements, at varying degrees (BERT embeddings at $r = 0.6435$, NumberBatch embeddings $r = 0.7393$, GPT-3 size 12288 at $r = 0.5071$, and GPT-3 size 1024 at $r = 0.5187$, with $p < .001$ for all). While the NumberBatch embeddings had the highest correlation with human judgements, these embeddings also performed better on this subset of 34 clue/board pairs than they did on the group of 400 boards (increasing from 66.25% to 70.59% of words guessed correctly), although this difference in performance on each of the sample of 34 clue/board pairs, and each of the total 400 clue/board pairs was not significant ($t(33) = 1.107$, $p = .2763$). Additionally, BERT and NumberBatch embedding similarity scores were more strongly correlated with each other than either approach was with human relatedness judgements ($r = 0.7449$, $p < .001$).

While the study discussed in the previous section and comparisons with Con-

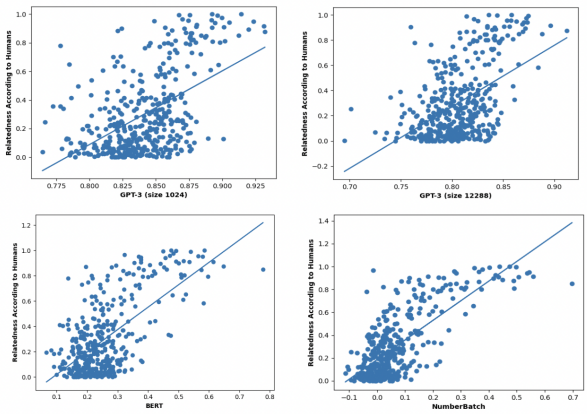

Figure 9.1: For each model indicated, embedding similarity scores are plotted against human relatedness judgements.

ceptNet failed to identify relationship types BERT based approach over or under emphasized, here we might characterize the success of such approaches at approximating human relatedness judgements at different levels of relatedness. In Figure 9.1, the similarity scores as calculated by embedding based approaches are plotted relative to human relatedness judgements. Plots look similar across models, which tend to underestimate word relatedness, but overestimate relatedness in the case of words that humans do not consider to be related. This trend can be visualized more clearly in the case of the BERT model in Figure 9.2.

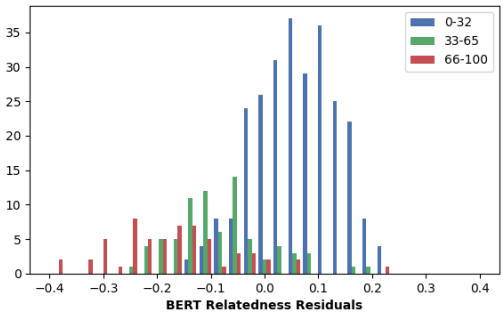

Figure 9.2: Residuals from BERT data shown in Figure 8.1. For 3 ranges of human relatedness judgements for word pairs (0-32,33-66,67-100), the difference between actual BERT relatedness judgements and BERT relatedness judgements predicted by regression. Positive residuals indicate BERT relatedness score is higher than predicted by regression.

Chapter 10

Conclusion

In this paper I have evaluated the ability of current NLP approaches to determine word relatedness in a human-like way in the game Codenames. Simple cosine similarity between word embeddings from large language models tended to perform best, and indeed captured many of the strengths of other approaches such as structured information from knowledge graphs and visual information from multi-modal embeddings. However, even the most successful approach did not exceed 80% of human performance on the task. Isolated human relatedness judgements performed just as well as human guessers on the task, suggesting that the difference in performance between humans and word embedding techniques is simply due to differences in word relatedness judgements. However, it was difficult to characterize just how these judgements differed. Labeling relationship types between clue/word pairs, either automatically through ConceptNet or through newly collected human explanations, did not reveal systematic differences in word relationships depending on how well word embeddings captured human relatedness judgements.

The findings discussed here do express a degree of optimism as to the ability of word embedding based approaches to learn structured knowledge about the world to a certain extent. They also express optimism about the theoretical quantifiability of

word relatedness. However, it is interesting that by far the largest model, GPT-3, did not outperform smaller models such as BERT or NumberBatch, especially given GPT-3's groundbreaking ability to generate human like text and do zero, one, and few shot learning. This indicates that language generation ability is not a good proxy for language understanding. Additionally, larger models do not necessarily learn more human-like language representations. To achieve such representations, we must consider the training data, goals, and architecture of word embedding models, rather than simply scaling these models up or feeding them more data.

Section 10.1

Future Directions

While the results in Chapter 8 suggest that it is indeed possible to explicitly map out relatedness between words based on isolated human judgements, doing so on a large scale would be extremely labor intensive and require endless updating. Additionally, simply having access to word relatedness judgements is not sufficient for most NLP tasks. While the accuracy of a more general purpose model's relatedness judgements can indicate the extent to which that model has a human-like understanding of language, these judgements are not the only important aspect of language understanding, and should be considered a symptom of a good model rather than the goal. Discussed below are features of more general purpose models which might gain an understanding of language such that they can better make such judgements.

In this paper I have discussed the limitations of models which learn only from the statistical patterns in text. Although the embedding-based approaches explored here did not benefit greatly from the incorporation of information from knowledge graphs or visual information from the Clip model, the fact remains that such models are trained to learn patterns which do not perfectly coincide with the patterns in

human experience or thought. The retrofitting process used to obtain NumberBatch embeddings (the most successful embeddings), in which word embeddings are biased towards embeddings for words that humans consider to be related, is one promising solution to this problem. In the case of NumberBatch embeddings, ConceptNet is the source of these human relatedness judgements. As discussed in Chapter 7, however, ConceptNet is by no means a complete reflection of human relatedness judgements. While continuing to build out tools such as ConceptNet may allow them to better supplement word embeddings, such a process requires lots of time and human effort. Embeddings might be similarly biased by emphasizing certain types of text during training. People are less likely to write about obvious item attributes or relations and everyday occurrences than they are to experience them, relative to other types of experiences. Text such as dictionaries which contain common knowledge about how words are related, procedural knowledge from sources such as how to manuals, or scripts or dialogues which detail everyday happenings and perceptions, might be emphasized during training to make up for this mismatch. This text could be repeated in the training data, or the learning rate might be increased when making predictions about such text.

It also might be worthwhile to store different types of knowledge in different ways. Human memory is sometimes divided into episodic memory, which stores experiences, procedural memory, which stores the steps for performing certain actions, and semantic memory, which stores factual knowledge. Language models might be supplemented by another network which is trained only on text sources containing more factual or procedural knowledge, such as dictionaries or how to manuals. These smaller networks might be updated more easily as such information changes (for example, when a new U.S. president is elected). The information contained in these networks might be used in different ways depending on the task. For the task of determining word

relatedness, a weighted average of the word embeddings produced by the different networks might produce more human-like word representations.

It is also important to consider how models interpret the input data they are exposed to. Although BERT embeddings are much smaller than GPT-3 embeddings, they outperform them on the guessing task, suggesting that they better approximate human word representations. One important difference between the two models is the fact that BERT is bidirectional, and takes into account the preceding and following context when making a prediction, while GPT-3 only takes into account the preceding context. While this makes BERT less equipped for text generation, it encourages it to incorporate more information about words into their embeddings. BERT embeddings for words must take into account the words that tend to come before and after them, while GPT-3 embeddings must only take into account the words that come before them. It would be interesting to see how a model as large and trained on as much data as GPT-3, but trained bidirectionally like BERT, would perform on the Codenames task. While we have seen that embedding size does not necessarily improve performance, given that GPT-3 has outperformed BERT on related NLP tasks, analysing such a model would further clarify the relative strengths and weaknesses of BERT and GPT-3 and allow them to be combined more optimally.

In chapter 5, I found that BERT's attention mechanism tended to bias word embeddings more towards unrelated contexts than towards contexts which might disambiguate their multiple meanings (for example, the embedding for "bank" was more similar to its embedding in "river bank" than "flavorless bank"). This differs from the way humans read text, during which our representations of words are most affected by contexts which relate to a particular sense of that word. By tweaking the training paradigm, we might cause the attention mechanism to behave in a more human-like way and favor word sense disambiguation over the encoding of unrelated

contexts when determining word representations. BERT is currently trained to do masked word prediction, and to predict whether or not two sentences occur successively. In both of these training paradigms, meanings of words are predicted by entire sentences, where representations for individual words in that sentence are averaged together. To make word sense disambiguation more worthwhile for these tasks, after creating the encodings for each token in a sentence and before averaging them to create a sentence embedding or masked word prediction, tokens which are most similar to another token in the sentence could be removed. In this paradigm, words which determine the meanings of other words are more likely to be removed than words that are more unrelated to the meaning of the sentence as a while, since these words will often have similar meanings to the words whose meaning they determine. In the sentence "I went to the river bank to sell armchairs," since "river" and "bank" often occur in similar contexts, the embedding for "river" might be removed. Then, "bank" would need to have incorporated the context of "river" in order to make a successful prediction about a masked word or next sentence. Thus, it would be more worthwhile to attend to contexts which determine the meanings of polysemous words in a sentence when producing the embeddings for those words.

The ideas discussed here aim to make up for statistical mismatches between human experiences and the descriptions of these experiences in training text, distinguish between different types of textual information so that they might be learned or used in different ways and at different rates, combine the benefits of larger models with more sophisticated architectural elements such as bidirectionality, and tweak training paradigms to encourage language models to encode aspects of language that humans consider to be important, such as word sense disambiguation. These research directions are united by the common goal of creating NLP techniques which have a more human-like understanding of language. While these are by no means the only

approaches that might be taken to accomplish this goal, the results presented in this paper suggest that they are promising next steps.

Bibliography

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, *Language models are few-shot learners*, Advances in Neural Information Processing Systems **33** (2020), 1877–1901.

[2] Robert Dale, *Gpt-3: What's it good for?*, Natural Language Engineering **27** (2021), 113–118.

[3] Simon De Dayne, Danielle Navarro, Amy Perfors, Marc Brysbaert, and Gert Storms, *The "small world of words" english word association norms for over 12,000 cue words*, Behavioral Research Methods **51** (2019), 987–1006.

[4] Jean-Benoit Delbrouck, Stephane Dupont Dupont, and Omar Seddati, *Visually grounded word embeddings and richer visual features for improving multimodal neural machine translation*, 2017 International Workshop on Grounding Language Understanding, ISCA, 2017.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, Proceedings of NAACL-HLT (2018), 4171–4186.

[6] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith, *Retrofitting word vectors to semantic lexicons*, Proceedings of NAACL, 2015.

[7] Christiane Fellbaum, *Wordnet: An electronic lexical database*, Bradford Books, 1998.

[8] Luciano Floridi and Massimo Chiriatti, *Gpt-3: Its nature, scope, limits, and consequences*, Minds and Machines **30** (2020), 681–694.

[9] Catalina Jaramillo, Megan Charity, Rodrigo Canaan, and Julian Togelius, *Word autobots: Using transformers for word association in the game codenames*, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **16** (2020), 231–237.

[10] Y. N. Kenett, E. Levi, D. Anaki, and M. Faust, *The semantic distance task: Quantifying semantic distance with semantic network path length*, Journal of Experimental Psychology: Learning, Memory, and Cognition **43** (2017), no. 9, 1470–1489.

[11] Andrew Kim, Maxim Ruzmaykin, Aaron Truong, and Adam Summerville, *Cooperation and codenames:understanding natural language processing via codenames*, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **15** (2019), 160–166.

[12] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel, *Unifying visual-semantic embeddings with multimodal neural language models*, arXiv preprint arXiv:1411.2539 (2014).

[13] Divya Koyyalagunta, Anna Sun, Rachel Lea, and Cynthia Rudin, *Playing code-names with language graphs and word embeddings*, Journal of Artificial Intelligence Research **71** (2021), 319–346.

[14] Abhilasha Kumar, Mark Steyvers, and David Balota, *Semantic memory search and retrieval in a novel cooperative word game: A comparison of associative and distributional semantic models*, Cognitive Science **45** (2021), no. 10, e13053.

[15] Y. Li, Z.A. Bandar, and D. Mclean, *An approach for measuring semantic similarity between words using multiple information sources*, IEEE Transactions on Knowledge and Data Engineering **15** (2003), no. 4, 871–882.

[16] Hongru Liang, Wenqiang Lei, Jun Wang, Adam Jatowt, and Zhenglu Yang, *From unimodal to multimodal word embeddings: A survey*, 05 2021.

[17] Edward Loper and Steven Bird, *Nltk: The natural language toolkit*, CoRR **cs.CL/0205028** (2002).

[18] Junhua Mao, Jiajing Xu, Kevin Jing, and Alan L Yuille, *Training and evaluating multimodal word embeddings with large-scale web annotated images*, Advances in neural information processing systems **29** (2016).

[19] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew Gray, William Brockman, The Google Books Team, Joseph Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin Nowak, and Erez Lieberman Aiden, *Quantitative analysis of culture using millions of digitized books*, Science (2010).

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean, *Distributed representations of words and phrases and their compositionality*, Advances in Neural Information Processing Systems **26** (2013), 3111–3119.

[21] Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi, et al., *Wordnet:: Similarity-measuring the relatedness of concepts.*, AAAI, vol. 4, 2004, pp. 25–29.

[22] Jeffery Pennington, Richard Socher, and Christopher Manning, *Glove: Global vectors for word representation*, Empirical Methods in Natural Language Processing (EMNLP) (2014), 1532–1543.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al., *Learning transferable visual models from natural language supervision*, International Conference on Machine Learning, PMLR, 2021, pp. 8748–8763.

[24] Radim Rehurek and Petr Sojka, *Gensim–python framework for vector space modelling*, NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic **3** (2011), no. 2.

[25] Nils Reimers and Iryna Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 11 2019.

[26] Sebastian Ruder, *Tracking progress in natural language processing.*

[27] Judy Shen, Matthias Hofer, Bjarke Felbo, and Roger Levy, *Comparing models of associative meaning: An empirical investigation of reference in simple language*

games, In Proceedings of the 22nd Conference on Computational Natural Language Learning, Association for Computational Linguistics, 2018, p. 292–301.

[28] Robyn Speer, Joshua Chin, and Catherine Havasi, *ConceptNet 5.5: An open multilingual graph of general knowledge*, 2017, pp. 4444–4451.

[29] Shao-Yen Tseng, Shrikanth Narayanan, and Panayiotis Georgiou, *Multimodal embeddings from language models for emotion recognition in the wild*, IEEE Signal Processing Letters **28** (2021), 608–612.

[30] Ganggao Zhu and Carlos A. Iglesias, *Computing semantic similarity of concepts in knowledge graphs*, IEEE Transactions on Knowledge and Data Engineering **29** (2017), no. 1, 72–85.