VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
«DEPARTMENT OF COMPUTATIONAL AND DATA MODELING»

Informaciniu technologijų 3 kurso kursinis darbas

Codenames Board Game Companion Using the SBERT Language Model

Stalo žaidimo "Codenames" kompanionas, naudojant SBERT kalbos modelį

Done by:
Arsenij Nikulin signature

Supervisor:
dr. Rokas Astrauskas

Vilnius
2025

Contents

Abstract

This paper presents an AI-driven companion for the cooperative board game *Codenames: Duet*, designed to generate and interpret clues using semantic vector representations. Leveraging the Sentence-BERT (SBERT) language model, the system computes cosine similarity between word embeddings to identify optimal clues and guesses while avoiding semantically dangerous associations. The application is implemented as a FastAPI backend with a modular JavaScript frontend, enabling real-time gameplay and visualization.

Two gameplay modes are supported: Player-Companion Mode, allowing live interaction between a human and the AI; and Companion-Companion Mode, which facilitates automated testing for parameter tuning and performance benchmarking. The system introduces configurable algorithms for clue safety filtering, multi-target grouping, and similarity-based word ranking. Existing approaches were analyzed, and enhancements were made in clue precision and interpretability.

This work contributes to the development of language-aware game agents and provides a controlled environment for evaluating NLP models in constrained, context-rich scenarios.

Santrauka

Stalo žaidimo "Codenames" kompanionas, naudojant SBERT kalbos modelį

Šis darbas pristato dirbtiniu intelektu paremtą pagalbininką bendradarbiavimo principu paremtam stalo žaidimui *Codenames: Duet*, skirtą užuominų generavimui ir interpretavimui naudojant semantines vektorių reprezentacijas. Pasitelkiant kalbos modelį Sentence-BERT (SBERT), sistema apskaičiuoja kosinuso panašumą tarp žodžių įterpinių (angl. embeddings), kad būtų galima rasti optimalias užuominas ir spėjimus, išvengiant semantiškai pavojingų asociacijų. Programėlė įgyvendinta naudojant FastAPI serverio dalyje bei moduliniu JavaScript pagrindu kliento sąsajoje, užtikrinant realaus laiko žaidimą ir vizualizaciją.

Palaikomi du žaidimo režimai: Žaidėjo-Pagalbininko režimas, leidžiantis žmogui tiesiogiai bendrauti su DI, ir Pagalbininko-Pagalbininko režimas, skirtas automatizuotam testavimui, parametrų derinimui ir našumo įvertinimui. Sistemoje įgyvendinti konfigūruojami algoritmai užuominų saugumo filtravimui, kelių taikinių grupavimui bei žodžių rikiavimui pagal semantinį panašumą. Buvo išanalizuoti esami sprendimai ir pagerintas užuominų tikslumas bei interpretavimo aiškumas.

Šis darbas prisideda prie kalbą suprantančių žaidimo agentų kūrimo bei suteikia kontroliuojamą terpę NLP modelių testavimui kontekstiškai turtingose situacijose.

Introduction

Using artificial intelligence (AI) in board games is becoming a useful way to test how well natural language processing (NLP) models and reasoning methods work in structured, goal-based settings. Unlike traditional strategy games like chess or Go, modern word games such as Codenames rely on language, ambiguity, and indirect clues.

This study introduces an AI companionfor the cooperative board game Codenames: Duet. It uses the Sentence-BERT (SBERT) model to create and understand word clues. SBERT turns words into vectors, allowing the system to measure how closely clue words match target words using cosine similarity. The AI's main goal is to find clues strongly related to the right words (agents) while avoiding clues linked to wrong ones (civilians and the assassin), creating a complex balancing task.

To design this system, we looked at other Codenames bots that used simple rules, word embeddings (like GloVe or Word2Vec), or hand-picked lists. Many of these systems were not flexible, didn't filter clues well, or couldn't handle fast decisions. Our approach improves on this by using clear steps for clue filtering based on similarity limits, scoring systems that reduce risky choices, and clue group generation using the most similar word options.

There are two ways to use the system: Player-Companion Mode for real-time play with a human, and Companion-Companion Mode for automated testing. The second mode helps test how different model versions, clue group sizes, similarity levels, and word sets affect performance.

In both modes, the AI uses a pipeline that includes generating word embeddings, filtering unsafe clues, and ranking clues by how well they match several targets. For guessing, the process runs backward-comparing the clue to all board words and picking those with the highest similarity that pass a set threshold. Everything is powered by a FastAPI backend and shown through a web interface built in JavaScript.

1 Background and Related Work

1.1 Artificial Intelligence in Board Games

When it comes to artificial intelligence in board games, the first thing that comes to mind is chess and the confrontation between humans and computers. Article [6] discusses how AI in chess has influenced the development of algorithms and understanding in the field of artificial intelligence, including areas not directly related to board games.

Today, humans no longer compete with AI in board games on an intellectual level but still continue to play not only against live opponents, but also against bots. Since the level of most players is inferior to the AI created by the developers, bots in such games often have an underestimated difficulty or are offered with different levels, from beginner to master. The online *chess.com* platform provides user-friendly opportunities to test yourself against bot [1].

Chess is one of the most popular games for testing various AI approaches due to its rich complexity and strategic depth. Unlike simpler games such as checkers [7] or tic-tac-toe, where the number of possible moves is limited and certain actions (like mandatory captures) are enforced, chess offers a vast decision space and long-term planning potential. Simpler games have already been solved, meaning that perfect play from both sides is known, making them unsuitable for exploring advanced AI techniques.

1.2 Overview of the Codenames Board Game

Codenames is a team association game in which players try to guess the code names of agents arranged on a 5×5 grid. One player in the team plays the role of a spymaster and sees a special key card showing which words belong to his team, which are neutral, which belong to the opponent, and where the killer word is located. The Spymaster can only give one word as a clue and one number indicating how many words on the field are related to that clue. The other players (agents) must, without asking questions, try to guess the correct words. The game requires a high level of associative thinking, the precision of the language, and the ability to consider possible associations with other people.

There are different versions of the game. **Codenames: Pictures** use abstract or meaningful images instead of words, making the association process more difficult and making the game especially interesting for visuals and children. Visual elements force players to think in images and look for unusual connections.

Codenames: Duet is a cooperative version of the game designed for two players. In this version, both players act as a spymaster and an agent. Each has his key map with the location of agents, and players take turns giving each other hints to find agents. Figure 1 demonstrates that the location of each word on the **board** is the same for each user. The difference is between different roles, which are represented by colors:

- **Agent** (green) - the goal of the game is to find all the representatives of this role. Player A must guess all of Player B's green cards and vice versa. In the original game, there are 9 for each player, but some of them are the same for both players, so there are 15 in total.
- **Assassin** (black) - guessing (revealing) words with this role instantly will lead to defeat.
- **Citizen** (gray) - neutral card, but does not reveal the target. Revealing this word leads to loss of a turn.

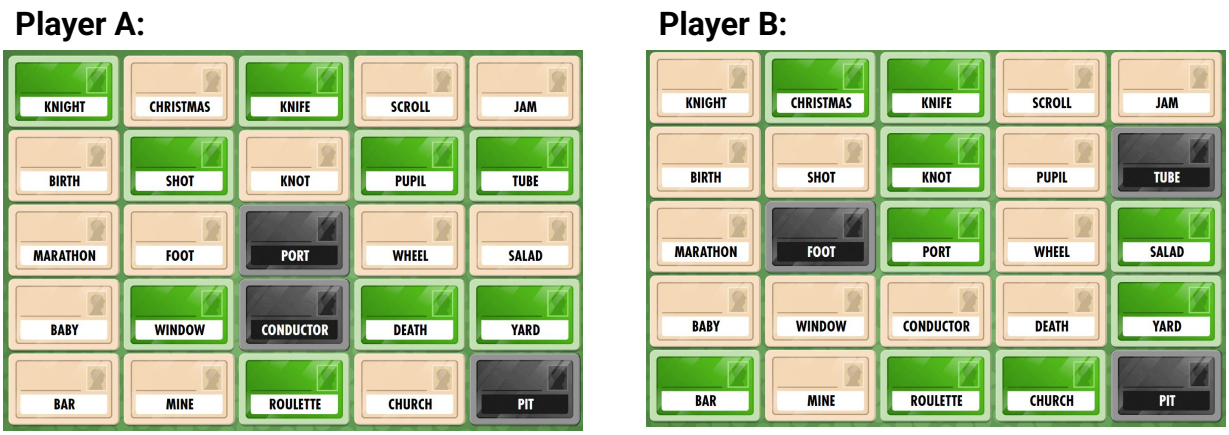

Figure 1. Player's board view

The only way for players to communicate is by giving **clues**. The clue consist of the **clue word** that connects words from the board, and **clue number** which indicates how many words connected. Example in figure 2 shows how player came up clue *criminal (3)* because he by this clue united 3 words *shot, knife, death* (with orange borders). Another player should reverse clue: based on clue *criminal (3)* find 3 most matching words.

Figure 2. Example of clue

To win the game players must guess all 15 agents for limited number of moves (represented by time tokens and in original game there is 9) and do not reveal any assassin card. Due limited time tokens, the game cycle presented in figure 3 completes $\frac{9}{2} = 4.5$ rounds. When time tokens runs out, players has last chance to guess all agents.

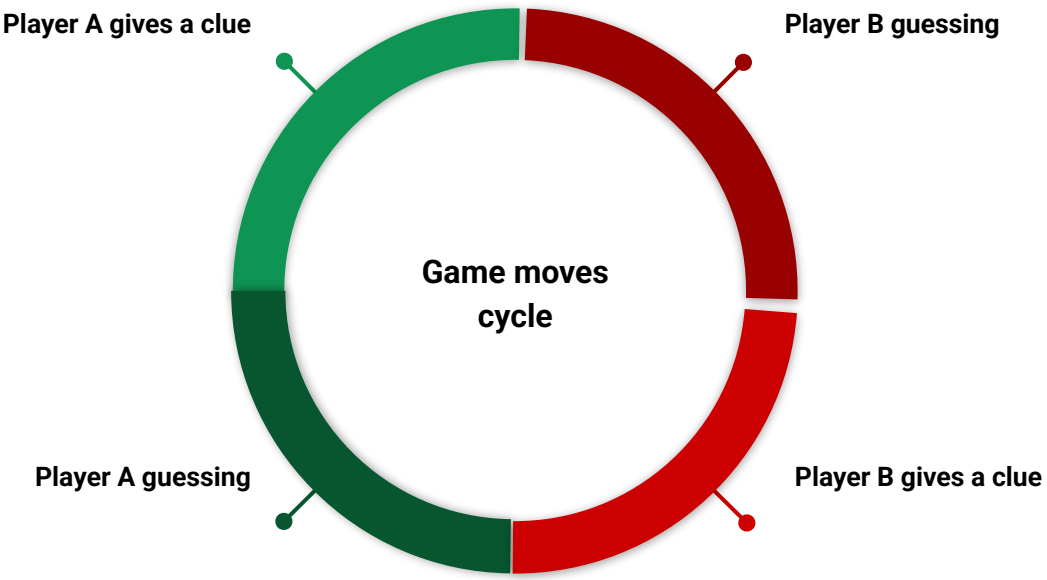

Figure 3. Codenames: Duet game cycle

An artificial intelligence companion was developed specifically for this version of the game. The goal was to create not only of guessing clues, but also of formulating them independently. In terms of the tasks facing the companion, the differences between the original version of Codenames and Duet are minimal: in both cases, it is necessary to understand the meaning of words, identify hidden connections between them, and avoid associations with *assassin* words. Therefore, the developed algorithms and approaches are applicable to both versions of the game with minor adaptations.

1.3 Natural Language Processing for Word Similarity

In the task of generating and interpreting cues in the game *Codenames*, the key element is the calculation of semantic proximity between words. For this purpose, Natural Language Processing (NLP) methods are used, which include conversion of words into vector representations. Such vectors can be obtained using the **Word2Vec**, **GloVe**, **FastText** or modern transformer models such as **BERT** and **Sentence-BERT**. These representations allow us to determine the degree of semantic relationship between words and select appropriate associative cues. There are other approaches, such as GPT, but it is too large to use for comparing individual words with each other.

To measure the proximity between vectors, **cosine similarity** is often used, calculated using the formula:

$$cosine\_similarity(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

where $\vec{a}$ and $\vec{b}$ are vectors of words, $\vec{a} \cdot \vec{b}$ is their scalar product, and $\|\vec{a}\|$ and $\|\|\vec{b}\|$ are the norms of the vectors (the length of the vector in Euclidean space). The value of cosine similarity ranges from -1 to 1, where 1 corresponds to complete semantic identity and 0 to complete lack of connection.

Thus, the use of NLP models and similarity metrics allows the AI companion to efficiently identify appropriate words for cues, minimizing the risk of selecting undesirable or dangerous associations.

1.4 Existing Approaches and Related Research

Research on AI agents for the game Codenames has emerged as a niche but insightful intersection of natural language processing (NLP), semantics, and game theory. Although the number of formal publications remains limited, several notable efforts have explored algorithmic approaches to clue generation and guessing, using a range of word representation techniques and learning paradigms. This section summarizes key existing approaches and highlights their methodological contributions and limitations.

A foundational study in this domain was presented at the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE) in 2019 [5]. This paper conducted a comparative analysis of different word embedding models, including Word2Vec and GloVe, across various dimensions (50d, 100d, 200d, and 300d). The researchers designed experiments in which AI agents played the role of spymasters and guessers, evaluating performance through game simulations. Their systematic approach and comprehensive results provide one of the earliest and most rigorous baselines for evaluating semantic models in the Codenames setting.

Another direction was explored in a 2022 arXiv preprint that applied reinforcement learning (RL) to the clue selection process [8]. The study proposed an environment where the agent learns optimal clue-giving strategies through reward-based interaction. Although the paper lacks clarity regarding the nature of its vocabulary and dataset (see section 3.1), it represents a promising step toward dynamic, trainable agents. In contrast to static word vector methods, reinforcement learning offers potential adaptability to various board configurations and player behaviors.

Several open-source projects have attempted to operationalize NLP-based agents for Codenames, though often without accompanying academic analysis. One such example is a 2022 GitHub project that aimed to compare Word2Vec, GloVe, and WordNet-based strategies [4]. Despite its conceptual relevance, the implementation is incomplete the dataset link is missing rendering it non-functional for reproduction or evaluation.

A similar effort from 2020 used GloVe embeddings to construct a simplified clue-generation system based on a trimmed vocabulary derived from common Codenames words [3]. While the project demonstrates practical implementation using GloVe's 300-dimensional vectors, it lacks documentation, logs, or evaluation metrics. Consequently, its scientific contribution is limited to anecdotal insights rather than reproducible experimentation.

An additional project from 2023 attempted to adapt AI gameplay for the cooperative variant Codenames: Duet [2]. While no paper accompanies the implementation, the platform supports interactive logging, and its adaptation to the Duet ruleset marks a rare case of applying AI in a two-player cooperative context. However, the lack of information about the underlying word dataset and clue-selection logic limits its evaluative and comparative potential.

In conclusion, existing research and prototypes reflect a growing interest in AI-based reasoning and communication within Codenames-like settings. However, many current systems suffer from issues of reproducibility, incomplete implementation, or absent datasets. Notably, few approaches target the cooperative Duet variant, despite its unique structural and cognitive challenges. This project aims to fill that gap by creating a fully functioning AI companion capable of both guessing and generating clues, with traceable behavior and semantic awareness grounded in modern NLP techniques.

2 System Design and Algorithms

2.1 System Requirements

The software being developed is a server application that processes HTTP requests in JSON format and uses the SBERT (Sentence-BERT) model to calculate semantic proximity between words. The requirements for the system are divided into functional and non-functional.

- Based on an input list of words with roles, the application must generate an appropriate associative clue.
- The returned result should contain a sorted list of inferred words according to their semantic proximity to the given clue.
- One of the SBERT models should be used to convert the words into a vector representation.
- Hints that turn out to be too close in meaning to the killer words should be excluded from the output.
- HTTP requests with a body in JSON format containing a list of words and additional information are accepted.
- Interaction with the application is carried out via REST API, and all responses are provided in JSON format.
- Response time when generating a clue or guessing words should not exceed 60 seconds under standard load.

2.2 Technologies and Tools

Created SBERT-based companion for Codenames: Duet game and visualization done on the following technologies and tools:

- **Python 3.12**: a programming language used to implement the game's server logic, AI companion logic, data processing, and interaction with the language model.
- **JavaScript**: used to create visualization and an interactive user interface. It displays the game field, tracks game status, and enables real-time user interaction with the application.
- **FastAPI**: a modern web framework in Python that provides API functionality. FastAPI processes requests from the front-end, launches game modes, and transfers data between the server and the client.
- **NumPy**: a library for numerical computations, used to work with vector representations of words.
- **SBERT model (Sentence-BERT)**: a language model based on BERT, designed to generate dense vector representations of words and phrases. SBERT is used in this work to evaluate the semantic proximity between hints and words on the game board. Each word is associated with a fixed-length vector.
- **Word lists**: pre-prepared lists of English words of varying length and frequency of use. Used as candidates for generating clues.

2.3 Clue Generation Algorithm

In this work, we propose an algorithm for generating clues in the game *Codenames: Duet* based on a given list of words and their assigned roles. The AI companion must suggest a clue that semantically connects as many target (agent) words as possible, while avoiding associations with assassin and civilian words.

Input and Output Data

Input: A list of words provided by the player, where each word $word_i$ is assigned a role $role_i$ from the AI companion's perspective. The role can be one of three types: *agent* (a target word), *citizen* (a neutral word), or *assassin* (a word that, if guessed, results in immediate loss).

$$word\_roles = \{(word_i, role_i) \mid word_i \in \text{String},\ role_i \in \{agent, citizen, assassin\}\}$$

Output: A clue word $clue\_word$ that semantically connects a selected group of target words. The result includes this generated clue, a group of words $(word_j, role_j)$ linked to the clue along with their semantic similarity scores $similarity_j$, and the total similarity score for the group.

$$(clue\_word,\ group,\ group\_similarity) \quad \text{where} \quad group = \{(word_j, similarity_j)\},\ similarity_j \in \mathbb{R}$$

Clue Generation Algorithm

Clue generation process is illustrated in the flowchart in Figure 4.

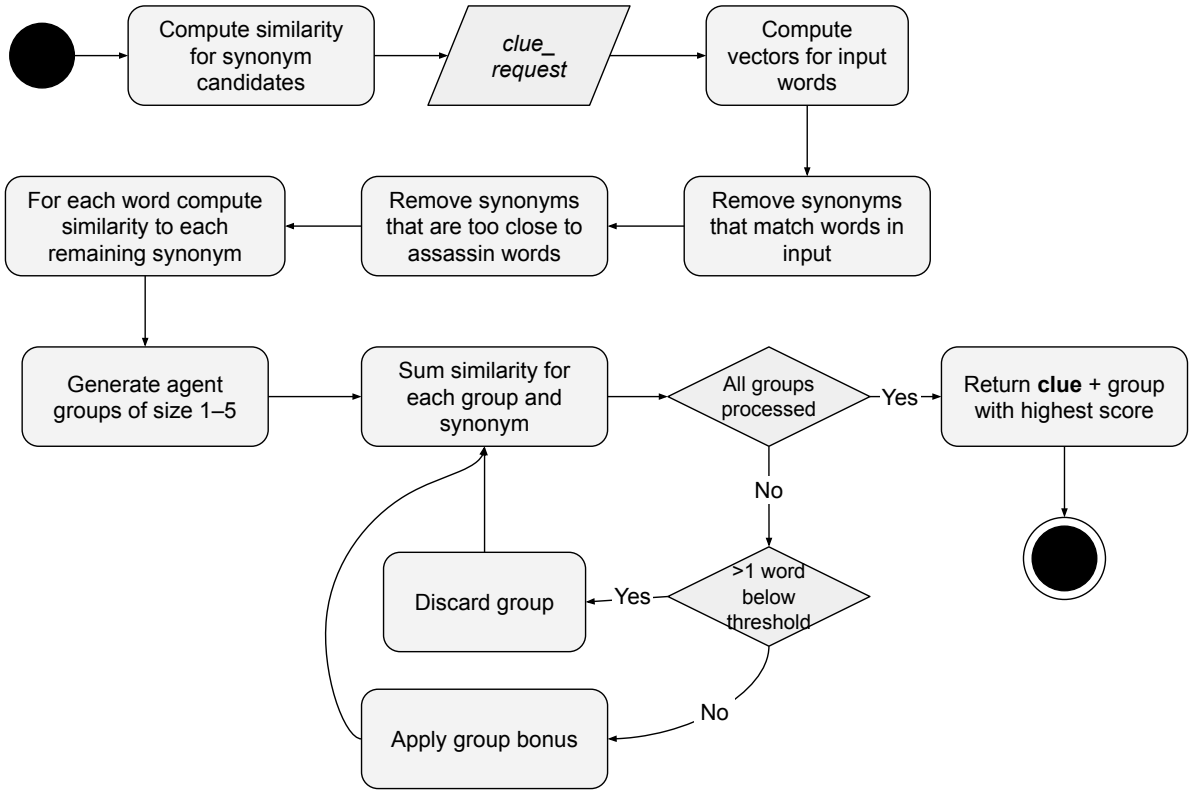

Figure 4. Flowchart of Clue Generation Algorithm

1. **Input and initialization:** The algorithm receives a list of word-role pairs:

$$clue\_request = [(word_1, role_1), \ldots, (word_n, role_n)]$$

   Each word is embedded using the SBERT model, resulting in a dictionary:

$$word\_vectors = \{WordRole : np.ndarray\}$$

2. **Filtering synonym candidates:** A predefined list of common English words is loaded with their vector embeddings.

   (a) Remove exact matches: If a synonym candidate is equal (case-insensitive) to a word in the input, it is excluded.

   (b) Remove unsafe synonyms: For each assassin word in the input, cosine similarity is computed with all synonym candidates. Synonyms with similarity above the threshold (> 0.45) are discarded.

3. **Computing similarity to valid synonyms:** For all remaining (non-assassin) words, the cosine similarity to each filtered synonym is computed. The results are stored as:

$$(word, synonym, similarity)$$

4. **Generating valid groups:** Groups of words are constructed for evaluation: agent-only groups of size 1 to 5.

5. **Building similarity lookup:** To speed up scoring, a lookup table is created:

$$similarity\_lookup[(word, synonym)] = similarity$$

6. **Evaluating all (group, synonym) pairs:** For each group and each synonym candidate. Similarity scores between each word and the synonym are summed, after that checked if more than one word has similarity below threshold, the total score is set to zero. Otherwise, a bonus proportional to the group size is added:

$$adjusted\_similarity = \sum_i similarity_i + 0.01 \cdot |Group|^2$$

7. **Selecting the best clue:** The algorithm returns the synonym with the highest adjusted score along with the associated group.

2.4 Word Guessing Algorithm

In addition to generating clues, the SBERT-based companion must also be capable of interpreting them. This section describes the algorithm used to process a given clue and return the most likely associated words based on semantic similarity.

Input and Output Data

Input: A triple consisting of: the clue word $clue\_word$, a number $clue\_number$ representing how many words the clue is intended to associate, and the list of non-revealed words $word\_list$ from the game board.

$$(clue\_word,\ clue\_number,\ word\_list)$$

Output: An ordered list of pairs where each $similarity_i$ represents the semantic similarity between the clue word $clue\_word$ and the candidate word $candidate\_word_i$, computed using cosine similarity. The list contains at most $clue\_number$ entries with $similarity_i > 0.45$.

$$[(candidate\_word_i,\ similarity_i)]$$

Guessing Algorithm Description

Algorithm flowchart diagram in figure 5.

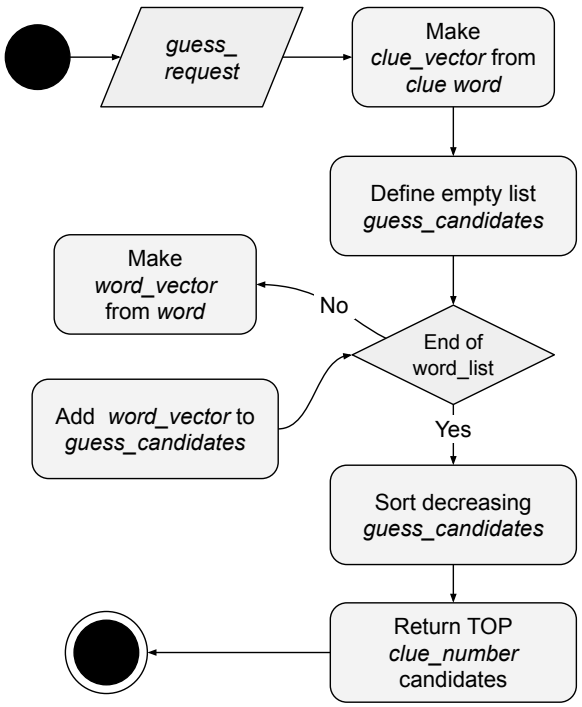

Figure 5. Flowchart of Guessing Algorithm

1. **Initialization:** The algorithm starts by extracting the three input fields from the request: the clue word, the desired number of guesses (clue number), and the list of candidate words (word list).

2. **Semantic similarity computation:** Each word in the word list is embedded using Sentence-BERT and compared to the clue word using cosine similarity. The resulting similarity scores are stored in a list of WordSimilarity objects.

$$WordSimilarity = (word,\ similarity)$$

3. **Ranking the guesses:** The list of word–similarity pairs is sorted in descending order based on their similarity scores.

4. **Selecting top guesses:** Words with similarity scores below a predefined threshold are removed. From the remaining list, the top clue number entries are returned as the AI's final guesses.

3 Companion Testing

3.1 Player-Companion Mode

The **Player-Companion Mode** is a cooperative gameplay mode in the *Codenames Duet AI* system, where user can in real time play with companion online to evaluate it's strength. Real-time communication is managed via a WebSocket connection. This allows instantaneous updates to the game board and history without requiring page reloads.

User Interface Components: Figure 6 illustrates screenshot of user interface of virtual board game.

- **clue-form**: A dynamic form shown when the human is expected to input a clue or see AI component's clue depends on game stage.
- **game-board**: A grid of clickable words that reflect the current visibility state and roles (hidden, agent, citizen, assassin).
- **game-history**: A chronological display of all clues and guesses with visual annotations. Hovering on AI clues and answers debug elements can be seen.
- **game-status** and **game-scoreboard**: Information about game cycle status (or game over status) and game progress.

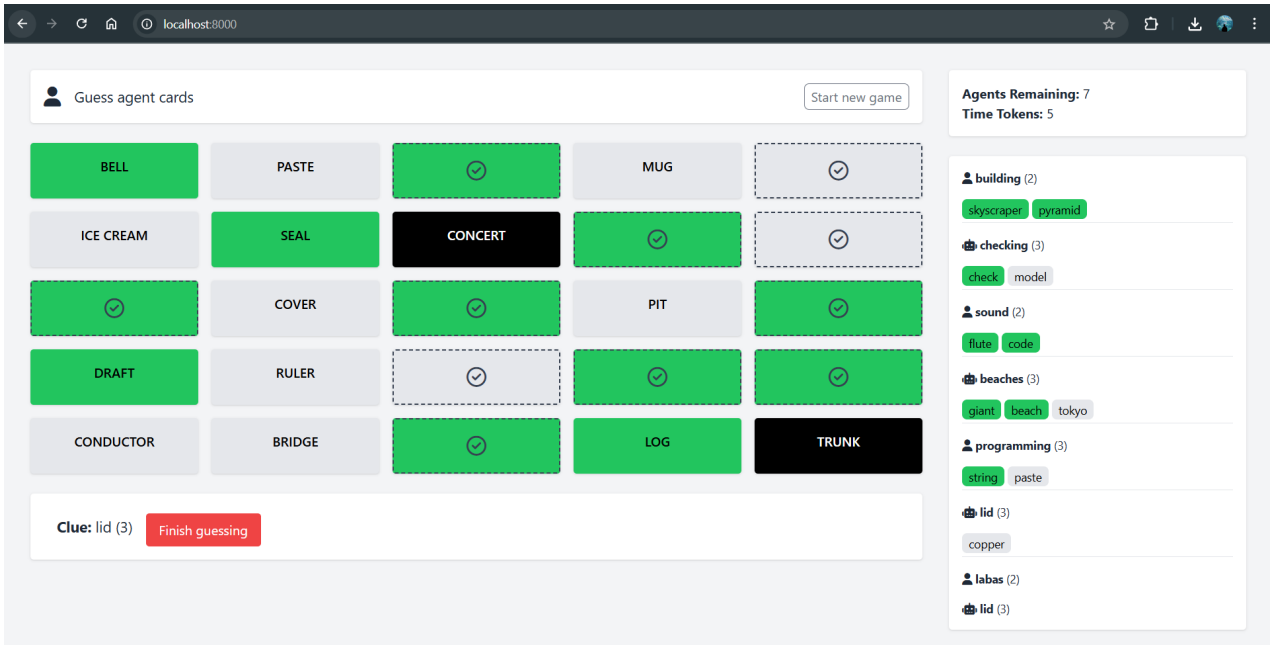

Figure 6. Virtual board game

3.2 Companion-Companion Mode

The Companion-Companion Mode is a fully automated testing configuration in the implemented Codenames Duet AI Companion system. In this mode, both the clue giver and guesser are controlled by the AI, allowing for rapid experimentation and benchmarking without human involvement.

Four main parameters influence the AI companion's behavior in this mode:

1. **Model version**: Determines the specific Sentence-BERT (SBERT) model used for computing semantic similarity between the clue and candidate words.
2. **Maximum group size**: Defines the largest number of words the AI is allowed to associate with a single clue. Also processing time depends on size. For example group if this maximum group size is 5 then total amount of groups - 391, for number 4 is 255.
3. **Threshold**: Sets the minimum similarity score required for a word to be considered a valid guess. The smaller this number, the more difficult it is for a person to understand AI.
4. **Words file**: The source list of candidate words available on the board, which affects the semantic space in which the AI operates. In this work used 10^3 and 10^4 word list lengths. Also content of this files also important.

This mode was primarily developed to support automated testing and evaluation of different configurations. The key metrics gathered during tests include:

- Total runtime or processing time
- Number of agents successfully guessed
- Final game result (win or timeout)

Although this mode does not fully simulate real human interaction, and its evaluation may be biased due to the use of the same model for both encoding and decoding (clue generation and interpretation with the same SBERT model), it remains a valuable tool. All gathered statistic about self testing here A.

Conclusions and Recommendations

This study introduced and implemented an AI-based companion for the board game *Codenames: Duet*, using the Sentence-BERT (SBERT) language model. The system was designed to handle both clue generation and word guessing, based on semantic vector proximity using cosine similarity.

Two main interaction modes were developed: *Player-Companion Mode*, enabling live interaction between a human and the AI agent; and *Companion-Companion Mode*, designed for automated system benchmarking. The latter was especially effective for analyzing the effects of key parameters such as SBERT model version, clue group size, similarity threshold, and word list selection.

Among the functional goals fulfilled by the system are: the ability to generate clues from a list of input words with roles, the ability to return a list of guesses based on semantic similarity, and the ability to filter clues that are semantically too close to assassin words. The architecture also supports future extensibility, such as integrating for original version of Codenames.

The solution was developed as a server-side application using Python 3.12 and the FastAPI web framework. The frontend was implemented using JavaScript and HTML, enabling real-time UI updates via WebSocket. The application adheres to requirements.

References

[1] chess.com. https://www.chess.com/play/computer.

[2] Codenames duet companion. https://word-association-ai.vercel.app/.

[3] Pbatch - codenames. https://github.com/Pbatch/Codenames.

[4] Xueweiyan - codenames-game-ai. https://github.com/XueweiYan/codenames-game-ai.

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

[6] Shiva Maharaj, Nick Polson, and Alex Turk. Chess ai: Competing paradigms for machine intelligence. *Entropy*, 24(4), 2022.

[7] Jonathan Schaeffer and Robert Lake. Solving the game of checkers. *Games of no chance*, 29:119–133, 1996.

[8] Sherman Siu. Towards automating codenames spymasters with deep reinforcement learning. *arXiv preprint arXiv:2212.14104*, 2022.

Appendices

A Testing Results

Different word list files

Parameter	Average Result After 5 Games
Time	57.9
Success	0
Agent Guessed	13.4
Time Tokens Remain	0

Table 1. Word list 10,000

Parameter	Average Result After 5 Games
Time	10.8
Success	0
Agent Guessed	12.6
Time Tokens Remain	0

Table 2. Word list 1,000

Different threshold

Parameter	Average Result After 5 Games
Time	8.1
Success	0.2
Agent Guessed	13.8
Time Tokens Remain	0.4

Table 3. Threshold 0.35

Parameter	Average Result After 5 Games
Time	9.4
Success	0.6
Agent Guessed	14.4
Time Tokens Remain	0.8

Table 4. Threshold 0.4

Different maximum group size

Parameter	Average Result After 5 Games
Time	10.8
Success	0
Agent Guessed	12.6
Time Tokens Remain	0

Table 5. Threshold 0.45

Parameter	Average Result After 5 Games
Time	6.4
Success	0
Agent Guessed	12.8
Time Tokens Remain	0

Table 6. Maximum group size 3

Parameter	Average Result After 5 Games
Time	7.2
Success	0.4
Agent Guessed	13.6
Time Tokens Remain	0.4

Table 7. Maximum group size 4

Parameter	Average Result After 5 Games
Time	8.1
Success	0.2
Agent Guessed	13.8
Time Tokens Remain	0.4

Table 8. Maximum group size 5