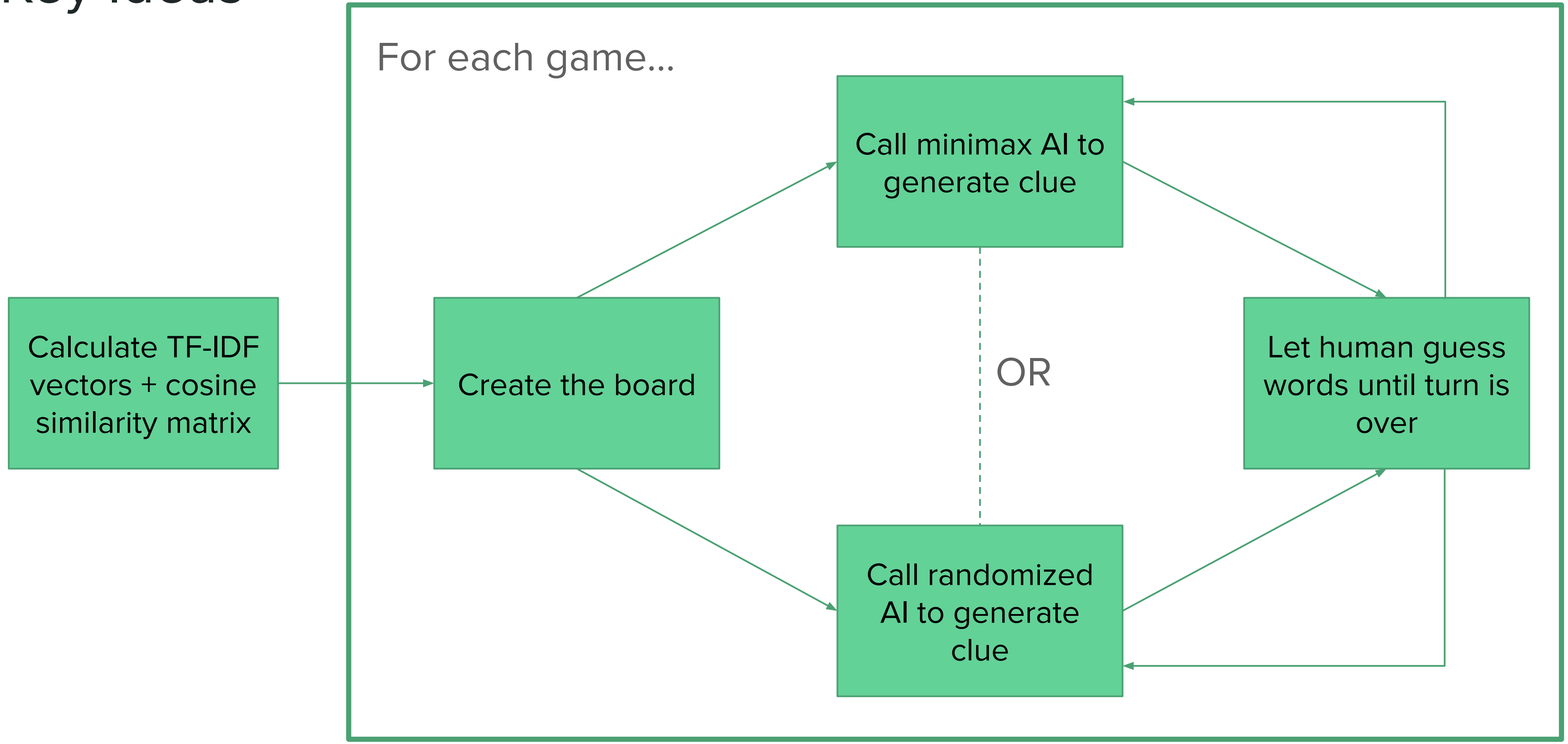
Codenames AI

Oleg Aprelikov, Shaurya Mathur, Aayushi Neema, Rohan Purandare

The Problem

- Existing spymaster algorithms
  - Are scarce
  - Try to maximize the number of words guessed per turn
  - Use standard word-embedding models
- Our goals
  - Create a spymaster AI using algorithms discussed in class
  - Design a word-association algorithm using Wikipedia articles
- Stretch goal
  - Create a field operative

Key Ideas


Calculate TF-IDF vectors + cosine similarity matrix
For each game...
Create the board
Call minimax AI to generate clue
OR
Call randomized AI to generate clue
Let human guess words until turn is over

Outcomes

- Word association
  - Inefficient scoring function: Unable to assign scores proportionally to clue words which statistically differentiate them from more common words. Therefore, cannot set benchmarks in code for quality check of guesses provided.
- Minimax AI
  - Often only provided clues to guess 1 word
  - Very slow due to size of TF-IDF vectors and some unnecessary computations
  - Plays defensively (goal is simply to play more efficiently than opponent)
- Randomized AI
  - Also often only provided clues to guess 1 word
  - In rare cases when there is a clue that targets more than 1 word, the AI often picks it even if it’s not very good

Challenges

- Spending up to 30 minutes generating all TF-IDF vectors
- For niche board-words, finding enough clue words with similarity scores above our threshold
- Eliminating common words / words that were semantically similar to board-words when coming up with clues
- Balancing weights of different word types (red, blue, neutral, assassin)

Q & A