0. Authors

Guapilla-Diaz, Javier; Zhang Ivonne

1. Agent & Persona Description

Agent Name: Dwight

More formally: **Dwight K. Scrute** (and **Dwight K. Scrute II** for the twin)

Our agent is based on Dwight Schrute. He is a confident, yet quirky character who tries to intimidate others, but doesn't always do so. He is very competitive and will criticize you for anything and everything. He can get aggressive, but is usually harmless.

2. Twin Agent

The twin agent isn't very different from the original, apart from the fact that it says it is Dwight Junior, and that it is the "clone of the clone." We tried looking for a ASCII art of Dwight Junior (which is an actual character in the Office, like season 9), but it was nowhere to be found. Therefore, play style is the same. Remarks, however, are different due to how we use AI to generate utterances, so things shouldnt be TOO repeated, even though we use the same prompt.

3. Implementation of Alpha-Beta Pruning

We based our AB pruning on the in-class slides provided, well we considered the Minimax function provided and added pruning on that, since it is simply an extension of Minimax. AB pruning itself was just a case where we kept track of some A and B and checked a>=b for max nodes or b<=a for min nodes, and then breaking if that happened as that would sort of mark the node as "done". While we haven't yet checked directly the performance boost of AB, time-wise wise it tends to run slightly faster, going down from 2-min for me (Javier) to about 1.5 min. For TTT, we have done the testing on make_move, and on average, our alpha-beta pruning uses 80% on average from the total static evals on all the levels. For FIAR, our alpha-beta pruning uses 94% on average from the total static evals on all the levels.

We did not implement the ordering of children by static eval.

4. Persona Description

We modeled our agent after Dwight K Schrute from The Office, as I, Javier, am I pretty big fan of the show. We considered other characters, but none of them seemed to have as much of a distinct personality/tone as Dwight. I also think he is one of the funnier characters. He “talks” with short sentences, but in the prompt, I tried to jam as much as I could to reflect what he might say. In the show, he is very competitive and threatening, so I tried to incorporate that without being too out of line, as well as over confident. Everyone hates Toby, so I made him the target of his insults. Although, I could’ve made it Jim, considering their rivalry.

5. Dialogue Relevance to State and Opponent

Our agent simply takes into consideration what the other agents last remark was. In our prompt, we just ask it to take it into account in their newly generated utter. It also considers the current score of the board, and attempts to tweak its response in accordance with that. We tried to make it so that our agent compares the opponent to Toby Flenderson, a character hated by almost everyone in The Office, or at least has lots of hate placed on him. Because of this, the agent will try to ridicule the opponent and make remarks about them, saying things like “thats what Toby would do.”

7. Developing the Dialog System

Initially, Ivonne made a bot-like response system, much like the one in random_player. We knew from the start that we wanted to use an LLM to generate our utterances, so it was just a start to figure out the functionality and how we would fit it into our scheme. We also weren’t sure at the time who we wanted to base our Agent on, so she made different types of utterances to account for that. We couldn't make too many comments about the current state of the board using the hard-coded bot-like utterances, so that is also why we wanted to use LLMs.

Once we got the LLM working, thanks to Ivonne, I tried to come up with a good prompt for the LLM call. I wanted to capture as much of Dwight as possible, so I actually went through a lot of different prompts until I got something I liked. We tried just making up scenarios and gameplay and made a prompt based on that to test how much of Dwight's essence it captured.

Something we got out of this was realizing that we can’t just say “you are Dwight,” but instead we have to sculpt our prompt in a way that sort of accounts for what we want the LLM to generate in terms of a nice response. At one point, we had responses that were 3-4 scrolls to the left to read the utter, to which we ended up adding the “short responses” to our prompt.

8. Extra Credit Implemented

1. Custom static_eval() Function

We wrote our own static evaluation function instead of using the simple version from the starter code. This function looks at every row, column, and diagonal, and counts how many X’s or O’s appear in a row.

Longer streaks are worth more (using powers of 10), so the agent can recognize when a line is close to winning. As a result, its minimax search makes smarter decisions even with a small search depth.

2. LLM-Based Dialogue System

We integrated Google’s gemini-2.5-flash model so our agent can speak in character with our prompt instead of using only hard-coded lines. We tried to use OpenAI, but it wasnt working for us, so we turned to Gemini. Ivonne put her credit card into the Gemini API so that we wouldn’t worry about exceeding any limits. Of course, it was just to use the free trial and then hit cancel in a week or two.

By using Gemini, we made sure to not make responses too long, and to keep them relevant and consistent to the character of Dwight.