A Behavior-Reactive Autonomous System to Identify Pokémon Characters

Xu Cao
Bohan Zhang
Jeremy Straub
Eunjin Kim
Introduction

Pokémon is an entertainment franchise with a large fan base. This project uses well-known Pokémon characters to demonstrate the operations of a question selection system. Presented in the form of a game where the computer attempts to guess the user-selected character, the system attempts to minimize the number of questions required for this purpose by identifying questions that most constrain the decision space. The decision making process is refined based on actual user behavior.

Methods

1. Algorithm
Roughly speaking, our system works as a high dimensional form of binary search. Since our goal is to guess user-selected character, in the ideal situation, we would be able to rule out half the remaining characters with user’s yes-no-answer to one question. Then we could narrow it down to the one right answer from around 2^N possibilities in N questions. The actual algorithm our system uses may be a little different from a typical binary search tree but in any case the goal is definitely to divide the set of possibilities as close to in half as possible with each question and user’s answer.

2. Data Base:
Our data base consists of 2 parts: Pokémon’s information and the questions to identify every Pokémon. And we will create 2 separate txt files to store these data when we run the program. As you can see in Figure 1, every Pokémon’s name is followed by an array of 0s and 1s which keeps track of what our system thinks the correct answer (an array match) is to all of the questions in the other txt file. If there are N characters and M questions, we’re storing N*M answers here.

3. Minimize the number of questions
Since the system’s goal is to ask as few questions as possible to get the correct answer, there must be an element of randomness and accuracy in choosing the questions. Our solution is to store a “popularity value” from previous users’ answers instead of only boolean values in the txt data file. This provides several advantages. First, “popularity value” makes it possible that not every answer has to apply to every character so that questions that don’t really apply to a character can have a value for the specific character close to .5 in the database. Second, people might disagree on some of the answers, so some questions might be more or less reliable in ruling out certain characters. Third, it gives our system more robustness against incorrect answers. When it remembers how often people guess each character and combines that knowledge with these vague answers, it can have some idea when it’s more likely that someone was guessing a popular character and gave an incorrect answer or when it’s more likely someone is trying to guess a character it’s never seen before.

Figure 1: Data file for Pokémon information

Conclusion

- After several tests, we project that our system can get the right answer 90% of the time.
- Our system is universal. As long as we have the right data base, our system could identify other franchise characters or even names under certain category.
- “Popularity value” could improve and grow our data base over time. Based on these values we can even generate new valuable questions.

Discussion

- We could be able to find a way to add a new feature to our system which displays previous users’ favorite Pokémon every time when a new user plays the game. This might have a psychology impact on the user so that he or she would pick a different character which would help us eliminate wrong guess.
- We could be able to find a way to provide users not only boolean answers but multiply choice answers like “yes/probably/don’t know/probably not/no answers”. Then our system could store some integer in the interval [0,1]. So if yes = 1, probably = .75, don’t know = .5, probably not = .25, and no = 0, we can basically store the expected value of a user’s answer to improve our system’s searching performance.

Acknowledgement

Research reported in this poster was supported by the Department of Computer Science, University of North Dakota.