Akinator Backend Logic and Machine Learning Design

Anirudh Malla

October 2025

Abstract

This document details the design and internal logic of the Akinator backend system, which performs probabilistic reasoning to identify a character through a sequence of user responses. The document also explains the challenges arising from dataset redundancy—where characters share highly similar traits—and describes the integration of fuzzy logic to improve uncertainty handling in user answers.

1 Introduction

The Akinator backend emulates an interactive guessing system that identifies a target character based on the user's answers to a series of questions. Each question corresponds to a *trait*, and each trait has one or more possible values.

The system maintains a probability distribution over all characters and updates it after each answer using Bayesian inference. It then selects the next question by maximizing expected information gain (entropy reduction).

2 System Architecture

The architecture consists of multiple layers designed for modularity and scalability:

- **Controllers:** Handle requests and session management.
- **Services:** Contain the core reasoning logic—probability updates, entropy computation, question selection, and result ranking.
- **Database Layer:** Provides access to characters, traits, and user response logs through the Prisma ORM.
- **Learning Module:** Continuously adjusts trait weights (information value and popularity) based on user feedback.

3 Core Probabilistic Model

3.1 Bayesian Update

Let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of all characters and $T = \{t_1, t_2, \ldots, t_m\}$ be the set of traits. After observing a user answer a for trait t_k, the probability of each character is updated as:

$$P(c_i|a, t_k) = \frac{P(a|c_i, t_k)\, P(c_i)}{\sum_j P(a|c_j, t_k)\, P(c_j)}$$

where $P(a|c_i, t_k)$ is the likelihood that character c_i would give answer a to trait t_k.

3.2 Entropy and Information Gain

Entropy quantifies the uncertainty in the character probabilities:

$$H(P) = -\sum_i P(c_i) \log_2 P(c_i)$$

For each candidate trait t_k, the expected entropy after asking about it is computed across all possible answers. The difference between the current and expected entropy yields the *information gain*:

$$IG(t_k) = H(P) - \sum_{a \in A_k} P(a) \cdot H(P|a, t_k)$$

The next question chosen is the one that maximizes $IG(t_k)$.

4 Dataset Challenge: Trait Redundancy

A major limitation observed in practice is that many characters share a large number of identical traits. For example, in a dataset containing hundreds of superheroes, traits like *"Super Strength"*, *"Human"*, or *"Male"* appear across dozens of entities.

This creates two core problems:

1. **Low Discriminative Power:** When many characters share the same trait values, each new answer reduces entropy only slightly. The information gain becomes nearly flat across traits, leading to poor question selection.

2. **Probability Convergence:** Due to overlapping traits, probabilities cluster together, preventing the model from achieving a strong confidence threshold for any single character.

In such a dataset, purely Boolean reasoning (*YES/NO*) is insufficient to capture subtle distinctions, motivating the introduction of fuzzy logic.

5 Fuzzy Logic Extension

To handle uncertainty and approximate reasoning, a fuzzy logic layer was introduced into the likelihood computation. Instead of mapping user answers directly to discrete values (*TRUE/FALSE/UNKNOWN*), each answer is assigned a degree of truth in the continuous range $[0, 1]$.

5.1 Fuzzy Normalization

Each user answer a is normalized into fuzzy values:

$$f(a) = \begin{cases} 1.0 & \text{for “YES” or “TRUE”} \\ 0.75 & \text{for “PROBABLY” or “LIKELY”} \\ 0.5 & \text{for “UNKNOWN” or neutral responses} \\ 0.25 & \text{for “PROBABLY\_NOT” or “UNLIKELY”} \\ 0.0 & \text{for “NO” or “FALSE”} \end{cases}$$

5.2 Fuzzy Likelihood Function

Given a character's stored trait value $v_c \in [0, 1]$ and a user response value $v_u = f(a)$, the fuzzy likelihood is defined using a Gaussian similarity function:

$$P(a|c_i, t_k) = e^{-\alpha(v_c - v_u)^2}$$

where α is a sensitivity constant (typically between 2 and 5) controlling how sharply the likelihood decreases as the difference between values grows.

This approach allows “probably” and “maybe” responses to partially support or contradict a character, rather than causing an all-or-nothing probability update.

5.3 Effect on Reasoning

Integrating fuzzy logic improves:

- Robustness to ambiguous or uncertain answers.
- Gradual probability convergence, rather than abrupt drops.
- Differentiation between characters sharing similar but not identical traits.

6 Algorithm Flow

1. Initialize all character probabilities equally.
2. Ask the question (trait) with highest information gain.
3. Fuzzify the user's answer and compute fuzzy likelihoods.
4. Update character probabilities using the fuzzy Bayesian update.
5. Compute entropy; if entropy or confidence meets stopping criteria, make a guess.
6. Otherwise, repeat with a new question.

7 Performance Observations

While fuzzy reasoning improves flexibility and stability, its effectiveness still depends on trait diversity. If multiple characters remain indistinguishable even after several questions, the algorithm tends to oscillate among them. This is an inherent limitation of datasets with low trait uniqueness, not of the inference model itself. Improvements that could have been implemented.

- Adaptive likelihoods learned from aggregate user data.
- Dynamic question generation using natural language models.
- Personalization of priors based on player history.

.