

# Assignment 4: Guess the Word

AI-2 Systems Project (Summer Semester 2026)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

Topic: Information gain/decision trees

Due on: August 14, 2026

Version from: June 19, 2026

Author: Jan Frederik Schaefer

**Make sure you sign up before working on this assignment.<sup>a</sup>**

**Using someone else's solution code, even as inspiration, is not allowed!**

**Sharing your solution code with other AISysProj students is not allowed.**

<sup>a</sup>You can still decide to postpone the assignment. Signing up includes an eligibility check, which avoids situations where you invest work into an assignment that you are not supposed to take.

## 1 Task summary

Implement an agent for a word guessing game. You can test your agent by competing on the server.

### Didactic objectives

1. Gain hands-on experience with decision-making based on information gain,
2. solve various challenges involving probabilistic reasoning.

### Prerequisites and useful methods

1. Basics of working with probabilities,
2. information gain and its use in making decision trees.

Note that the problem does not exactly match the definition of decision trees as discussed in the lecture. Nevertheless, understanding how information gain is used in decision trees should help with this assignment.

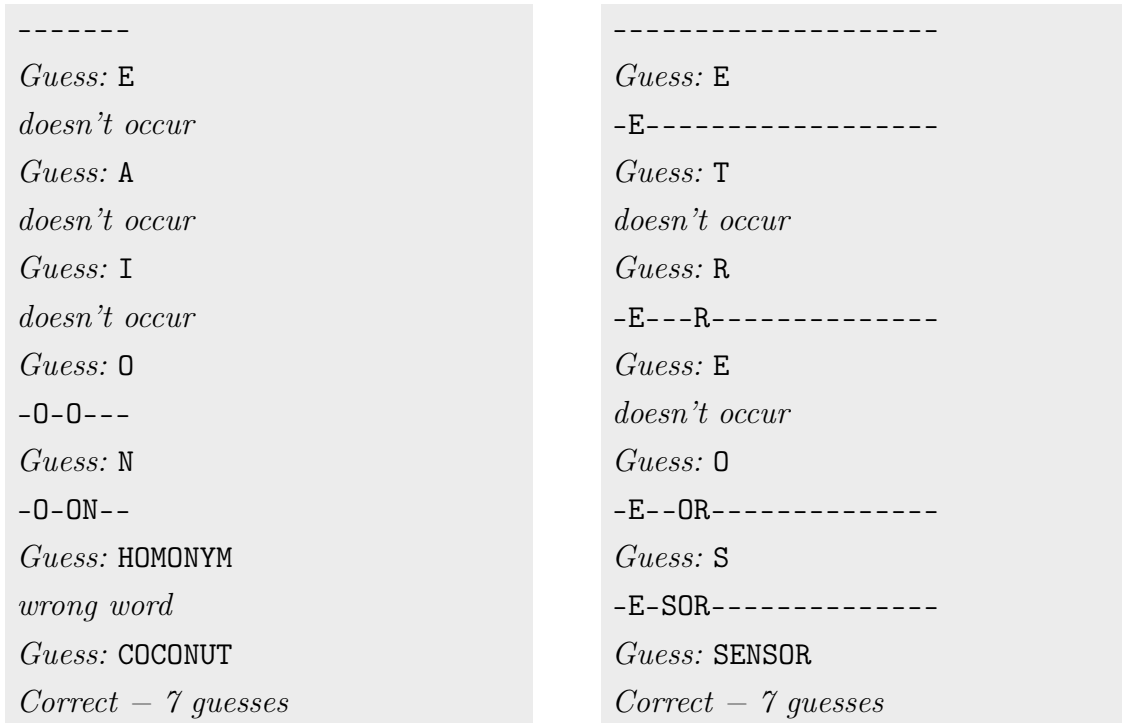


Figure 1: Example games with the standard rules (left) and the advanced rules (right).

## 2 Rules

The rules are inspired by the popular word-guessing game hangman [WH]. The player has to guess a word by repeatedly making letter guesses or word guesses. After each guess, the player gets feedback (e.g. about the positions of the guessed letter). The goal is to use as few guesses as possible to find the right word. In this assignment we will use two different sets of rules: *standard rules* and *advanced rules*.

### 2.1 Standard rules

The word is initially represented by dashes, where each dash replaces a letter of the word. If the player makes a letter guess, all occurrences of that letter in the word are revealed. The game ends if the player correctly guesses the word. See Figure 1 (left) for an example game.

### 2.2 Advanced rules

There are two changes in the advanced rule:

- Only a single letter location is revealed at a time (it is picked uniformly randomly among all unrevealed positions of that letter).
- The length of the word is disguised by appending more dashes in the feedback.

Figure 1 (right) shows an example game with the advanced rules.

## 2.3 How the word is chosen

The words are picked from a list of cities from [WCD]. The assignment repository [AR] contains a copy of that list (you should use the copy because the original might get modified). Concretely, the values from the `city_ascii` column are used, but only cities with a population (column `population`) of at least  $10^5$  are considered. The cities are chosen in a biased manner: the probability that a particular city is chosen is proportional to its population. City names that contain diacritics, hyphens or spaces are ignored.

## 3 Guessing words on the server

You should test and evaluate your agent by competing on the server. The rating of your agent is the lowest average number of guesses in 1000 consecutive games – the lower your rating, the better. Your agent can communicate with the server via HTTP requests. A Python implementation of the protocol is provided in the assignment repository. That means that you just have to implement the agent function – all the server interaction is already implemented for you.

The details of the protocol are described in [CSP], but you will only need them if you want to create your own implementation (e.g. in a different programming language). Reach out if you need help with that.

### 3.1 Action requests

The server will send you action requests, which contain an identifier for the request and a JSON object describing the current state of your word-guessing endeavour. Here is an example state description:

```
{
  "feedback": "-O-ON--",
  "guesses": ["E", "A", "I", "O", "N", "HOMONYM"]
}
```

## 3.2 Sending Actions

Your agent should respond to an action request by sending an action as a string:

1. For word guesses, the agent should send the word (in capital letters) as a string.
2. For letter guesses, the agent should send the letter as a string (again capitalized). As there are no single-letter words in this game, the server can distinguish letter guesses from word guesses.

## 4 What to submit

Your solution should be submitted to your team's repository. It should contain:

1. All your code for solving this assignment.
2. A README.md file explaining
  - i. dependencies (programming language, version, external libraries and how to get them),
  - ii. how to run your code,
  - iii. the repository structure,
  - iv. anything else we should know.
3. A solution summary (see [\[SoS\]](#) for more details – it should describe the main ideas, not document the code).

## 5 Points

You can get up to 80 points for the rating (average number of guesses in 1000 games) of your agent according to the server (assuming it is reproducible). Concretely, you will get the following points for the standard rules:

- 10 points if the rating is  $\leq 10$ .
- 20 points if the rating is  $\leq 7.5$ .
- 30 points if the rating is  $\leq 5.5$ .
- 35 points if the rating is  $\leq 4.8$ .
- 40 points if the rating is  $\leq 4.4$ .

For the advanced rules, you will get additionally

- 10 points if the rating is  $\leq 13$ .
- 20 points if the rating is  $\leq 9.5$ .

- 30 points if the rating is  $\leq 8.5$ .
- 35 points if the rating is  $\leq 7.5$ .
- 40 points if the rating is  $\leq 6.5$ .

Assuming you have at least a partial solution, you can additionally get up to 20 points for the quality of the submission (README, solution summary, ...). The maximum number of points is therefore 100. If the grading scheme doesn't seem to work well, we might adjust it later on (likely in your favor).

## References

- [AR] *Repository for Assignment 4: Guess the Word*. URL: <https://gitlab.rrze.fau.de/wrv/AISysProj/ss26/a2.4-guess-the-word/assignment>.
- [CSP] Jan Frederik Schaefer. *AISysProj server – Clients and server protocol*. URL: <https://aisysprojserver.readthedocs.io/en/latest/clients.html>.
- [SoS] *Solution Summary*. URL: <https://gitlab.rrze.fau.de/wrv/AISysProj/admin/general/-/blob/main/solution-summary.md>.
- [WCD] *World Cities Database*. URL: <https://simplemaps.com/data/world-cities> (visited on 01/07/2025).
- [WH] *Hangman (game)*. URL: [https://en.wikipedia.org/wiki/Hangman\\_\(game\)](https://en.wikipedia.org/wiki/Hangman_(game)) (visited on 07/27/2022).