Sudoku Dataset in .json format

Bill Xiao

2023-08-01

Package

```
library(jsonlite)
library(sudoku)
```

Get an object

```
get_level <- function(givens) {
  if (givens > 46) {
    level <- "Effortless"
  } else if (givens >= 36 && givens <= 46) {
    level <- "Easy"
  } else if (givens >= 32 && givens <= 35) {
    level <- "Medium"
  } else if (givens >= 28 && givens <= 31) {
    level <- "Difficult"
  } else if (givens >= 17 && givens <= 27) {
    level <- "Evil"
  } else {
    level <- "Unsolvable"
  }
  return(level)
}


# j = random_seed
puzzle <- function(j, number_of_blanks, print_puzzle = FALSE, print_solution = FALSE) {
  set.seed(j)
  givens = 81-number_of_blanks
  # Generate a valid solved Sudoku grid
  solved_grid <- generateSudoku(Nblank=0, print.it=F)

  level = get_level(givens)

  # Create a puzzle by removing specified number of cells
  filled_cells <- which(solved_grid != 0)
  cells_to_remove <- sample(filled_cells, number_of_blanks, replace = FALSE)
  puzzle_grid <- solved_grid
```

```
  puzzle_grid[cells_to_remove] <- 0

  puzzle_matrix = matrix(puzzle_grid, nrow = 9, ncol = 9, byrow = TRUE)
  solution_matrix = matrix(solved_grid, nrow = 9, ncol = 9, byrow = TRUE)
  # Print the puzzle and solution if specified
  if (print_puzzle) {
    print("Question:")
    print(puzzle_matrix)
  }

  if (print_solution) {
    print("Solution:")
    print(solution_matrix)
  }

  json_data <- list(
    Id = sprintf("%02d%03d", number_of_blanks, j), # number_of_blanks(2) + seed(3)
    Question = puzzle_matrix,
    Answer = solution_matrix,
    Trial = 0,
    Level = level, #("Effortless",...,"Evil")
    Shot = 0, # (0,1,2)
    Model = NULL, # ("GPT3","GPT3.5","GPT4")
    Error = NULL  # ("NO", "Hallucination", "Violation")

  )

  return(json_data)

}

# Example
# generated_puzzle <- puzzle(42, 40, print_puzzle = TRUE, print_solution = TRUE)
# This will generate a puzzle with random_seed = 42 and number_of_blanks = 40
```

Write on .json (verification)

```
pr = 0.5 # prob of a correct answer

veri <- function(pr = 0.5, seed1, seed2) {
  if (pr < 0 || pr > 1) {
    stop("Probability 'prob' must be between 0 and 1.")
  }
  flag = sample(c(0,1),1,replace=F,prob=c(1-pr,pr)) # 1 stands for correct answer
  # create an initial board
  initial_board <- puzzle(seed1+seed2, 0, print_puzzle = F, print_solution = F)$Answer

  if (flag==1){
    num_changes = 0
    json_data <- list(
    Id = sprintf("%03d%03d", seed1, seed2), # 6-digit, eg., 021089
```

```
    Question = initial_board,
    Answer = initial_board,
    Trial = 0,
    Change = num_changes,
    Shot = 0, # (0,1,2)
    Model = NULL, # ("GPT3","GPT3.5","GPT4")
    Error = NULL  # ("NO", "Hallucination", "Violation")
    )
  }
  else{
    # seed1 for getting the number of places to be changed (1,2,3)
    set.seed(seed1)
    num_changes <- sample(1:3, 1)
    # seed2 for getting the position of places to be changed
    set.seed(seed2)
    change_coords <- sample(1:81, num_changes, replace = FALSE)
    # action
    modified_board <- initial_board
    for (coord in change_coords) {
      possible_values <- setdiff(1:9, modified_board[coord])
      new_value <- sample(possible_values, 1)
      modified_board[coord] <- new_value
    }
    json_data <- list(
    Id = sprintf("%03d%03d", seed1, seed2), # 6-digit, eg., 021089
    Question = modified_board,
    Answer = initial_board,
    Trial = 0,
    Change = num_changes,
    Shot = 0, # (0,1,2)
    Model = NULL, # ("GPT3","GPT3.5","GPT4")
    Error = NULL  # ("NO", "Hallucination", "Violation")
    )
  }

  return(json_data)
}

# example
# new_sudoku <- veri(pr = 0, seed1 = 12, seed2 = 53)
# print(new_sudoku)

# To generate .json file
veri_json_file <- function(pr=0.5, Number, seed1_start, seed2_start) {
  json_data_list <- list()
    for (i in 1:Number) {
      for (j in seed1_start:(Number+seed1_start)) {
        object <- veri(pr, seed1 = j, seed2 = seed2_start+j)
        json_data_list <- c(json_data_list, list(object))
      }
    }
  json_file <- sprintf("Verification_%03dNumber_%02fProb_%03dSeed%03dSeed",
                       Number, pr, seed1_start, seed2_start)
```

```
  writeLines(toJSON(json_data_list, pretty = TRUE), json_file)
}


# Parameters are set manually
Number <- 100
seed1_start <- 100  # lower bound
seed2_start <- 111  # upper bound

# Example
# veri_json_file(pr=0.5, Number, seed1_start, seed2_start)
# This will generate a "verification" dataset of 100 samples, with probability of a correct board being
```

Write on .json (solution)

```
# To generate .json file
solu_json_file <- function(K, N1, N2) {
  json_data_list <- list()
    for (number_of_blanks in N1:N2) {
      for (random_seed in 1:K) {
        json_data = puzzle(random_seed, number_of_blanks)
        json_data_list <- c(json_data_list, list(json_data))
      }
    }
  json_file <- sprintf("Sudoku_%03dSeed_%02dLower%02dUpper", K, N1, N2)
  writeLines(toJSON(json_data_list, pretty = TRUE), json_file)
}


# Parameters
# random_seed from 1 to K
K <- 100
# Define the number of blanks - N1 to N2
N1 <- 10  # lower bound
N2 <- 64  # upper bound

# N2 <- 64  # largest possible upper bound = 81-17

# generate_json_file(K, N1, N2)

solu_json_file(K=30, N1=18, N2=36)
# This will generate a "Solution" dataset of 30 samples for every number_of_blanks ranging from 18 to 3
```