Triage Against the Machine: Can AI Reason Deliberatively?

Francesco Veri, Gustavo Umbelino

2025-03-21

Define functions

Maybe move this to it's own package...

```
create_file_path <- function(provider, model, survey, file_type) {
  file.path("llm_data", provider, model, survey, paste0(file_type, ".csv"))
}
```

Get available LLMs

```
# Read the CSV file into a data frame and remove duplicates
models <- read_csv("private/llms_v2.csv", show_col_types = FALSE) %>%
  distinct(provider, model)

# Initialize a vector to store the 'has_data' values
has_data_flags <- logical(nrow(models))

# Iterate over each row in the models data frame
for (i in 1:nrow(models)) {
  provider <- models$provider[i]
  model <- models$model[i]

  # Create the path
  path <- paste0("llm_data/", provider, "/", model)

  # Check if the path exists and set the 'has_data' flag accordingly
  has_data_flags[i] <- file.exists(path)
}

# Add the 'has_data' column to the models data frame
models <- models %>%
  mutate(has_data = has_data_flags)

# Print rows where has_data is TRUE
if (any(models$has_data)) {
  print(models %>% filter(has_data == TRUE))
} else {
  warn("No data available!")
}
```

```
## # A tibble: 15 x 3
##    provider  model                has_data
##    <chr>     <chr>                <lgl>
```

```
##  1 google    gemini-1.5-pro       TRUE
##  2 google    gemini-2.0-flash     TRUE
##  3 google    gemini-1.5-flash     TRUE
##  4 google    gemini-1.5-flash-8b  TRUE
##  5 google    gemma2               TRUE
##  6 meta      llama3.2             TRUE
##  7 microsoft phi4                 TRUE
##  8 mistralai mistral-large-latest TRUE
##  9 mistralai ministral-8b-latest  TRUE
## 10 mistralai mistral-small-latest TRUE
## 11 mistralai open-mistral-nemo    TRUE
## 12 mistralai open-mixtral-8x22b   TRUE
## 13 openai    gpt-4o               TRUE
## 14 openai    o1-mini              TRUE
## 15 openai    gpt-3.5-turbo        TRUE
```

Get available surveys

```r
# Read the sheet names of the Excel file
survey_names <- excel_sheets(SURVEY_FILE)

# remove invalid and "template"
survey_names <- survey_names[!grepl("^~", survey_names) & survey_names != "template"]

print(survey_names)
```

```
##  [1] "uppsala_speaks"      "fnqcj"               "acp"
##  [4] "ccps"                "forestera"           "biobanking_mayo_ubc"
##  [7] "zh_uster"            "zh_thalwil"          "zh_winterthur"
## [10] "ds_bellinzona"       "ds_aargau"           "fremantle"
## [13] "zukunft"             "bep"                 "energy_futures"
## [16] "valsamoggia"         "gbr"                 "auscj"
## [19] "swiss_health"        "biobanking_wa"
```

```r
# Define the file types
file_types <- c("considerations", "policies", "reasons")
```

Read and format LLM data

```r
# initialize an empty list to store the data frames
data_list <- list()
index <- 0

# iterate over each survey
for (survey_name in survey_names) {

  # iterate over each row in the models data frame where has_data is TRUE
  for (i in 1:nrow(models)) {
    if (models$has_data[i]) {
      provider <- models$provider[i]
      model <- models$model[i]

      # check if any file for the survey exists
      survey_path <- paste0("llm_data/", provider, "/", model, "/", survey_name, "/")
```

```r
    if (!any(file.exists(paste0(survey_path, file_types, ".csv")))) {
      next
    }

    # Iterate over each file type
    for (file_type in file_types) {
      # Create the file path
      file_path <- create_file_path(provider, model, survey_name, file_type)
      index <- index + 1

      # Check if the file exists
      if (file.exists(file_path)) {
        # Read the CSV file
        temp_data <- read_csv(file_path, show_col_types = FALSE)

        # Skip file if file exists but has no data
        if (nrow(temp_data) == 0) {
          warn(paste0(file_path, " exists but has no data!"))
          break
        }

        meta <- c(
          "cuid",
          "created_at",
          "provider",
          "model",
          "temperature",
          "input_tokens",
          "output_tokens"
        )

        # Select the relevant columns based on file type
        if (file_type == "considerations") {
          survey_data <- temp_data %>%
            rename_with( ~ paste0("C", seq_along(.)),
                         starts_with("C", ignore.case = FALSE))

          # add column "survey" to meta data
          survey_data <- survey_data %>%
            mutate(survey = survey_name) %>%
            relocate(survey, .after = model)
          meta <- c(meta, "survey")

          # Ensure survey_data has columns up to C50
          for (j in (ncol(survey_data) - length(meta) + 1):50) {
            survey_data[[paste0("C", j)]] <- as.numeric(NA)
          }

          # go to next file type
          next

        } else if (file_type == "policies") {
          temp_data <- temp_data %>%
```

```
              select(cuid, starts_with("P", ignore.case = FALSE)) %>%
              rename_with( ~ paste0("P", seq_along(.)),
                           starts_with("P", ignore.case = FALSE))

            # Ensure temp_data has columns up to C50
            for (j in (ncol(temp_data)):10) {
              temp_data[[paste0("P", j)]] <- as.numeric(NA)
            }

          } else if (file_type == "reasons") {
            temp_data <- temp_data %>%
              select(cuid, reason) %>%
              rename(R = reason)
          }

          # merge the data frames by 'cuid' and keep all rows
          survey_data <- full_join(survey_data, temp_data, by = c("cuid"))

        }
      }

      # Add the survey_data data frame to the list
      if (exists("survey_data")) {
        data_list[[length(data_list) + 1]] <- survey_data

        # Remove the survey_data data frame to free up memory
        rm(survey_data)
      }

    }
  }
}
```

```
## Warning: llm_data/openai/o1-mini/uppsala_speaks/considerations.csv exists but
## has no data!

## Warning: llm_data/openai/o1-mini/fnqcj/considerations.csv exists but has no
## data!

## Warning: llm_data/openai/o1-mini/acp/considerations.csv exists but has no data!

## Warning: llm_data/openai/o1-mini/ccps/considerations.csv exists but has no
## data!

## Warning: llm_data/openai/o1-mini/forestera/considerations.csv exists but has no
## data!

## Warning:
## llm_data/mistralai/mistral-large-latest/biobanking_mayo_ubc/considerations.csv
## exists but has no data!

## Warning: llm_data/openai/o1-mini/biobanking_mayo_ubc/considerations.csv exists
## but has no data!

## Warning: llm_data/openai/o1-mini/zh_uster/considerations.csv exists but has no
## data!

## Warning: llm_data/openai/o1-mini/zh_thalwil/considerations.csv exists but has
```

```
## no data!
```

```
## Warning:
## llm_data/mistralai/mistral-small-latest/zh_winterthur/considerations.csv exists
## but has no data!
```

```
## Warning: llm_data/openai/o1-mini/zh_winterthur/considerations.csv exists but
## has no data!
```

```
## Warning: llm_data/openai/o1-mini/ds_bellinzona/considerations.csv exists but
## has no data!
```

```
## Warning: llm_data/openai/o1-mini/ds_aargau/considerations.csv exists but has no
## data!
```

```
## Warning: llm_data/mistralai/mistral-small-latest/fremantle/considerations.csv
## exists but has no data!
```

```
## Warning: llm_data/openai/o1-mini/fremantle/considerations.csv exists but has no
## data!
```

```
## Warning: llm_data/openai/o1-mini/zukunft/considerations.csv exists but has no
## data!
```

```
## Warning: llm_data/mistralai/mistral-large-latest/bep/considerations.csv exists
## but has no data!
```

```
## Warning: llm_data/openai/o1-mini/bep/considerations.csv exists but has no data!
```

```
## Warning: llm_data/openai/o1-mini/energy_futures/considerations.csv exists but
## has no data!
```

```
## Warning: llm_data/mistralai/mistral-large-latest/valsamoggia/considerations.csv
## exists but has no data!
```

```
## Warning: llm_data/openai/o1-mini/valsamoggia/considerations.csv exists but has
## no data!
```

```
## Warning: llm_data/mistralai/mistral-large-latest/gbr/considerations.csv exists
## but has no data!
```

```
## Warning: llm_data/openai/o1-mini/gbr/considerations.csv exists but has no data!
```

```
## Warning: llm_data/meta/llama3.2/auscj/considerations.csv exists but has no
## data!
```

```
## Warning: llm_data/openai/o1-mini/auscj/considerations.csv exists but has no
## data!
```

```
## Warning: llm_data/openai/o1-mini/swiss_health/considerations.csv exists but has
## no data!
```

```
## Warning: llm_data/openai/o1-mini/biobanking_wa/considerations.csv exists but
## has no data!
```

```
# Combine all data frames in the list into a single data frame
llm_data <- bind_rows(data_list)

write_csv(llm_data, paste(OUTPUT_DIR, "llm_data.csv", sep = "/"))

inp <- mean(llm_data$input_tokens)
outp <- mean(llm_data$output_tokens)
tot <- inp + outp
```

```r
# delete data_list from memory
rm(data_list)
rm(temp_data)

# Aggregate llm_data by provider, model, and survey and N the number of rows
llm_surveys <- llm_data %>%
  group_by(provider, model, survey) %>%
  summarise(
    N = n(),
    mean_input_tokens = as.integer(mean(input_tokens)),
    mean_output_tokens = as.integer(mean(output_tokens)),
    .groups = 'drop'
  )

cost_tokens <- llm_data %>%
  group_by(provider, model) %>%
  summarise(
    N = n(),
    input_tokens = as.integer(sum(input_tokens)),
    output_tokens = as.integer(sum(output_tokens)),
    .groups = 'drop'
  )

# Print the summary
print(head(llm_surveys))
```

```
## # A tibble: 6 x 6
##   provider model            survey        N mean_input_tokens mean_output_tokens
##   <chr>    <chr>            <chr>     <int>             <int>               <int>
## 1 google   gemini-1.5-flash acp          60              5244                 328
## 2 google   gemini-1.5-flash auscj        10              4601                 319
## 3 google   gemini-1.5-flash bep           9              4469                 301
## 4 google   gemini-1.5-flash biobanki~    60              3912                 276
## 5 google   gemini-1.5-flash biobanki~    10              5167                 346
## 6 google   gemini-1.5-flash ccps         57              3546                 246
```

```r
# write summary to file
write_csv(llm_surveys, paste(OUTPUT_DIR, "llm_surveys.csv", sep = "/"))
write_csv(cost_tokens, paste(OUTPUT_DIR, "cost_tokens.csv", sep = "/"))
```

Calculate Cronbach's Alpha

```r
# knitr::opts_chunk$set(echo = T, results = "hide")

# Initialize an empty list to store the alpha results
alpha_results <- list()

# Iterate over each unique provider/model combination
for (provider_model in unique(paste(llm_data$provider, llm_data$model, sep = "/"))) {
  # Filter the data for the current provider/model
  provider_model_data <- llm_data %>% filter(paste(provider, model, sep = "/") == provider_model)
```

```r
  # Iterate over each survey
  for (survey_name in unique(provider_model_data$survey)) {
    # Filter the data for the current survey
    survey_data <- provider_model_data %>% filter(survey == !!survey_name)

    # Calculate Cronbach's Alpha for considerations (C1..C50)
    considerations_data <- survey_data %>% select(starts_with("C", ignore.case = FALSE))

    if (nrow(considerations_data) > 1) {
      alpha_considerations <- alpha(considerations_data, check.keys = TRUE, warnings = FALSE)$total$raw
      #alpha_considerations <- round(alpha_considerations, 2)

    } else {
      alpha_considerations <- NA
    }

    # Calculate Cronbach's Alpha for policies (P1..P10)
    policies_data <- survey_data %>% select(starts_with("P", ignore.case = FALSE))

    if (nrow(policies_data) > 1) {
      alpha_policies <- alpha(policies_data, check.keys = TRUE, warnings = FALSE)$total$raw_alpha
      #alpha_policies <- round(alpha_policies, 2)
      } else {
      alpha_policies <- NA
    }

    # Store the results in the list
    alpha_results[[length(alpha_results) + 1]] <- tibble(
      provider_model = provider_model,
      survey = survey_name,
      N = nrow(considerations_data),
      alpha_considerations = alpha_considerations,
      alpha_policies = alpha_policies
    )
  }
}

# Combine all results into a single data frame
alpha_results <- bind_rows(alpha_results)

rm(considerations_data)
rm(survey_data)
rm(policies_data)
rm(provider_model_data)

# Print the results
print(head(alpha_results))
```

```
## # A tibble: 6 x 5
##   provider_model        survey             N alpha_considerations alpha_policies
##   <chr>                 <chr>          <int>                <dbl>          <dbl>
## 1 google/gemini-1.5-pro uppsala_speaks    10                0.935          0.649
## 2 google/gemini-1.5-pro fnqcj             10                0.909          0.761
## 3 google/gemini-1.5-pro acp               10                0.893          0.694
```

```
## 4 google/gemini-1.5-pro ccps              10                0.745          0.760
## 5 google/gemini-1.5-pro forestera         10                0.910          0.722
## 6 google/gemini-1.5-pro biobanking_ma~    10                0.830          0.637
```

```r
# write summary to file
write_csv(alpha_results, paste(OUTPUT_DIR, "alpha_results.csv", sep = "/"))

knitr::knit_exit()
```