The model is contained in various formats where the inputs are always the same, but the output may vary. The *Task Sheets* is the most straightforward version, here the tasks of an occupation(s) are used as an input, and the output is an excel file that contains: all the O*NET-SOC occupations sorted by similarity to the input occupation(s), the quantitative vector(s), and the mappings generated by the model for each input task. The *Similarity Results* produces only the list of occupations similar to the input occupation(s) ranked in descending order. Finally, the *Vectors of Occupations* code generates only the quantitative vectors for the input occupation(s).

For either of the three code versions for the model to run two inputs are needed, the O*NET data which has already been formatted to interact with the code (e.g. ONET Data (28.1).xlsx). Furthermore, the aggregation rule can be set to min, mean, or max using *"metric_to_use"*. Other changes in the model that can be implemented in the code include toggling skills, abilities, or knowledge off (meaning that they would not be part of the quantitative vector), and choosing a different similarity metric "*metric_type*", where "similarity" corresponds to cosine similarity and is covered on the paper while "threshold" is not.

To get started use the *Task Sheets* python script on your preferred IDE. Start an excel document and write the name of an occupation on the top row and its tasks on the following rows. To add more occupations follow the same nomenclature on the column N+1. Load the excel file with the occupation information and O*NET release file (e.g. O*NET Data 30) on the script (i.e. make sure they are in the same directory as the script). Execute the script and the output should be an excel file for each occupation that contains: all the O*NET-SOC occupations sorted by similarity to the input occupation(s), the quantitative vector(s), and the mappings generated by the model for each input task.

Notes:

- The *Vector of Occupations* script will automatically generate the vector for all aggregation rules where columns 4 and 5 correspond to min, 6 and 7 to max, and 8 and 9 to mean. When using the *Comparison of Vectors* script to compare to other vectors, it is necessary to make sure that the right "*metric_to_use"* is being used.
- Make sure that the name of the sheet which contains your occupations and tasks in the excel file is "*Sheet1*". Otherwise change it in the code to make sure it is pointing to the right direction *"sheet1_name"*.

1. **Evaluation A (Coherence)**

Inputs: ONET Data (28.1).xlsx; Tasks Organized by Occupations (Task Statements).xlsx
Code: Vectors of Occupations; Similarity Results
Outputs: Occupation Results (All) - Similarity Mean.xlsx;
Other: ONET (28.1) Occupations Model KSAO Vectors - Metrics

Use the Vectors of Occupations to generate the vectors for all O*NET-SOC occupations using only their tasks - *Tasks Organized by Occupations (Task Statements).xlsx*. Manually we calculate precision, recall, number of attributes with importance > "1", and importance ≥ "3", and percentage of attributes with sufficient level on the *ONET (28.1) Occupations Model KSAO Vectors - Metrics.xlsx*[1]. To obtain the sorted list of occupations similar to the model-generated vectors for O*NET occupations, we use the Similarity Results code (notice that those results can also be obtained using the *Vectors of Occupations* and *Comparison of Vectors* code).

2. b. Evaluation B (Congruence), Model Application for ARM Integrator, and ARM sensitivity

Inputs: ONET Data (28.1).xlsx; Tasks Organized by Occupations ARM.xlsx
Code: Task Sheets
Outputs: Task Sheets ARM Robotics {Technician, Specialist, Integrator}.xlsx

Use the *Task Sheets* script to generate the vectors for all ARM occupations and obtain the O*NET SOC occupations sorted by similarity. Change the metric rule *"metric_to_use"* to generate the quantitative vector under the min, mean, and max aggregation rules.

3. Model Application PERM

Inputs: ONET Data (30).xlsx; PERM_Disclosure_Data_FY2024_Q4_old_form [Raw].xlsx; PERM_Disclosure_Data_FY2024_Q4_old_form.xlsx; PERM Q4 2024 Input {(A), (B), (C), (D), (E), (F)}.xlsx; PERM Q4 2024 Output CLEANED {(A), (B), (C), (D), (E), (F)}.xlsx; ONET (30) KSAO All Data.xlsx
Code: PERM Cleanup [Br]; PERM Open AI Cleanup Parallelized; Vectors of Occupations; PERM Beta Distribution; PERM Random Uniform Distribution; PERM Empirical Distribution; PERM Metrics All - {Model Focus, Expert Focus, Expert Focus}
Outputs: ONET (30) KSAO Vector PERM Q4 2024 {A, B, C, D, E, F)} {Cleaned, Cleaned Synthetic Random, Cleaned Synthetic Empirical, Cleaned Synthetic Beta Dis}.xlsx; ONET (30) KSAO Vector PERM Q4 2024 (All) Cleaned - {Metrics, Synthetic Beta Dist, Synthetic Empirical, Synthetic Random} {A, EF, MF}.xlsx

First, we manually transpose and copy over relevant data from *PERM_Disclosure_Data_FY2024_Q4_old_form [Raw].xlsx* to *PERM_Disclosure_Data_FY2024_Q4_old_form.xlsx*. Then, we perform a quick manual cleanup of profiles that do not contain any information on the *"Specific Skills"*, or that contain the sole entry of *"Not Applicable, N/A, or None"* on the *PERM_Disclosure_Data_FY2024_Q4_old_form.xlsx* file. After that we separate profiles into batches of 5000 *PERM Q4 2024 Input {(A), (B), (C), (D), (E), (F)}.xlsx* to avoid hitting Excel column and row limits. Each of the input profiles is processed through the *PERM Cleanup [Br]*

[1] Data for the O*NET expert-generated vectors is found on the O*NET Data {28.1, 30, etc}

to generate the output profiles which separates individual tasks/skills from the "specific skills" section using break line *[br]* and *;* as signals into *PERM Q4 2024 Output {(A), (B), (C), (D), (E), (F)}.xlsx*. After the tasks have been separated, we proceed to using OpenAI's "GPT-4o-mini" to remove lines that do not contain substantive occupation-specific information this way we generate the PERM Q4 2024 Output CLEANED {(A), (B), (C), (D), (E), (F)}.xlsx; files. All these steps are implemented to batch the original PERM data, and remove non-information tasks and occupations.

Use the *Vectors of Occupations* script along with *ONET Data (30).xlsx* file to generate the *ONET (30) KSAO Vector PERM Q4 2024 CLEANED {A, B, C, D, E, F}.xlsx* vectors from the *cleaned output* batches. These are the model generated vectors for PERM occupations. To generate baseline vectors for those batches we use the PERM Beta Distribution; PERM Random Uniform Distribution; PERM Empirical Distribution scripts in which the inputs are the original model-generated PERM profiles *ONET (30) KSAO Vector PERM Q4 2024 {A, B, C, D, E, F}.xlsx* and the *ONET (30) KSAO All Data.xlsx* file. Each of the output files *ONET (30) KSAO Vector PERM Q4 2024 {A, B, C, D, E, F)} {Cleaned, Cleaned Synthetic Random, Cleaned Synthetic Empirical, Cleaned Synthetic Beta Dis}.xlsxt* are processed using the test scripts PERM Metrics All - {Model Focus, Expert Focus, Expert Focus} to produce the aggregate scores for share within standard deviation for the model and baselines; all vectors *{A, B, C, D, E, F)}* must be processed as a batch.

In the SI we show how the scores vary when re-running the OpenAI API and how the model performs when the profiles are not cleaned from strenuous information. That analysis can be performed by following the steps outlined above, but using as inputs the respective files (e.g. *ONET (30) KSAO Vector PERM Q4 2024.xlsx* (A), and *ONET (30) KSAO Vector PERM Q4 2024 (A2) Cleaned.xlsx*).

4. Attribute Vectors Sensitivity Under Different Aggregation Rules

Inputs: ONET Data (28.1).xlsx; Tasks Organized by Occupations (Task Statements).xlsx
Code: Similarity Results, Vector of Occupations
Outputs: Occupation Results (All) - Similarity {Mean, Min, Max}.xlsx;
Other: ONET (28.1) Occupations Model KSAO Vectors - Metrics

Use the *Vectors of Occupations* script to generate the vectors for all O*NET-SOC occupations using *Tasks Organized by Occupations (Task Statements).xlsx*, and changing the metric to min, mean, and max. Manually we calculate, precision, recall, number of attributes with importance > "1", and importance ≥ "3", and percentage of attributes with sufficient level on the *ONET (28.1) Occupations Model KSAO Vectors - Metrics.xlsx*. To obtain the sorted list of occupations similar to the model-generated vectors for O*NET occupations, we use the Similarity Results code (notice that those results can also be obtained using the *Vectors of Occupations* and *Comparison of Vectors* code).

5. **Attribute Vectors Sensitivity Under Different Aggregation Rules**

Inputs: ONET Data (28.1).xlsx; Tasks Organized by Occupations (Task Statements).xlsx
Code: Similarity Results - Top 3, Vector of Occupations
Outputs: Occupation Results (All) - Similarity {Mean, Min, Max}.xlsx;
Other: ONET (28.1) Occupations Model KSAO Vectors - Metrics Top 3 Distinct

Use the *Vectors of Occupations* script to generate the vectors for all O*NET-SOC occupations using *Tasks Organized by Occupations (Task Statements).xlsx*, and changing the metric to min, mean, and max. Manually we calculate, precision, recall, number of attributes with importance > "1", and importance ≥ "3", and percentage of attributes with sufficient level on the *ONET (28.1) Occupations Model KSAO Vectors - Metrics.xlsx*. To obtain the sorted list of occupations similar to the model-generated vectors for O*NET occupations, we use the Similarity Results code (notice that those results can also be obtained using the *Vectors of Occupations* and *Comparison of Vectors* code).

6. **Evaluation Across Releases**

Inputs: ONET Data (ONET Data 24.2 & 28.1).xlsx; Tasks Organized by Occupations 9 New with KSAO (Task Statements).xlsx
Code: Similarity Results
Outputs: Occupation Results (9 New Occupations with KSAO) - Similarity Mean.xlsx

Use the *Similarity Results* script to obtain the sorted list of occupations similar to the model-generated vectors for O*NET occupations, the output should be the list of O*NET-SOC occupations ranked by similarity to the 9 new occupations (when going from O*NET 24.2 to 28.1).

7. **OpenAI Simple Occupational Match**

Inputs: Tasks Organized by Occupations ARM.xlsx; Tasks Organized by Occupations (Task Statements).xlsx
Code: OpenAI - Simple Occupational Match (API)
Outputs: ARM Robotic Technician - Simple Occupational Match (API).xlsx

Use the *OpenAI - Simple Occupational Match (API)* script to obtain the sorted list of occupations similar to the input occupation; the input files are *Tasks Organized by Occupations ARM.xlsx; Tasks Organized by Occupations (Task Statements).xlsx*. This is a binary insert method which will take a long time to process. This method was chosen because it reduced hallucination by carefully attending three tasks at a time.

8. **OpenAI Embeddings and API for Translation**

Inputs: Tasks Organized by Occupations ARM.xlsx; Tasks Organized by Occupations (Task Statements).xlsx; ONET Data (28.1).xlsx
Code: OpenAI - Task Sheets (API) - Translation; OpenAI - Vectors of Occupations (Embeddings) - Translation; Comparison of Vectors
Outputs: Task Sheets ARM Robotics {Technician, Specialist, Integrator} - Mean (OpenAI API).xlsx
Other: ONET (28.1) Occupations Model KSAO Vectors - Similarity - OpenAI - Metrics.xlsx

Use the *OpenAI - Vectors of Occupations (Embeddings)* script to generate the occupation vector for all O*NET-SOC occupations (using *Tasks Organized by Occupations (Task Statements).xlsx; ONET Data (28.1).xlsx*), here the output will be *ONET (28.1) Occupations Model KSAO Vectors - Similarity - OpenAI.xlsx*. Use that vector file on the *Comparison of Vectors* script and obtain the sorted list of O*NET-SOC occupation vectors similar to the input (*Occupation Results (All) - Similarity Mean - OpenAI.xlsx)*. We manually calculate precision and recall on the *ONET (28.1) Occupations Model KSAO Vectors - Similarity - OpenAI Metrics.xlsx*

To test the API for translation, use the *OpenAI - Task Sheets (API) - Translation* script and inputs *Tasks Organized by Occupations ARM.xlsx, ONET Data (28.1).xlsx*. The output are *Task Sheets ARM Robotics {Technician, Specialist, Integrator} - Mean (OpenAI API).xlsx*

9. **OpenAI Embeddings and API for Translation**

Inputs: Tasks Organized by Occupations (Task Statements) - Random 50.xlsx; ONET Data (28.1).xlsx; Tasks Organized by Occupations ARM
Code: OpenAI - Vector of Occupations (API), Comparison of Vectors
Outputs: ONET (28.1) Occupations OpenAI (Def and Anch) KSAO Vectors Random 50.xlsx; Occupation Results OpenAI (Def and Anch) Random 50 - Similarity
Other: ONET (28.1) Occupations OpenAI (Def and Anch) KSAO Vectors Random 50 Metrics.xlsx

Use the *OpenAI - Vector of Occupations (API)* script to generate the occupation vector for all 50 random occupations (using *Tasks Organized by Occupations (Task Statements) - Random 50.xlsx; ONET Data (28.1).xlsx*), here the output will be *ONET (28.1) Occupations OpenAI (Def and Anch) KSAO Vectors Random 50.xlsx*. Use that file on the *Comparison of Vectors* script and obtain the sorted list of O*NET-SOC occupation vectors similar to the input (*Occupation Results (All) - Similarity Mean - OpenAI.xlsx*). We, manually calculate precision, recall, number of attributes with importance > “1”, and importance ≥ “3”, and percentage of attributes with sufficient level on *ONET (28.1) Occupations OpenAI (Def and Anch) KSAO Vectors Random 50 Metrics.xlsx* Finally, apply the same procedure above for the *Tasks Organized by Occupations ARM.xlsx* to generate the ARM vectors.

Note: When using the *OpenAI - Vector of Occupations (API)* script, it will only produce a singular vector (no min or max). Therefore, when using the *Comparison of Vectors* script make sure to use the “min” metric for the code to choose the appropriate columns (4 and 5) for the comparison.