# Roboface-Excel works this way:

It looks for folders called “config” and “Excels” at the same directory as the roboface-excel folder containing the executables.

1. It reads the config.ini file to get parameters (creativity and response length) and filepaths to the Client Answer Library named file.
2. It takes a user input for the Excel filename and constructs the full file path.
3. The script loads the Excel file and defines a function **is\_question(sentence)** to identify if a given cell contains a question. Criteria:
   1. Begins with question words like "what", "where", "when", "why", "how", "which", "who", "whom", or "whose".
   2. Contains "you" or "your".
   3. Has a question mark (?).
   4. Starts with the word "please".
   5. Begins with words like "provide" or "describe".
   6. Begins with the phrase "Name of".
4. It creates a DataFrame to store sheet names, cell locations, questions, and answers.
5. It loops through each sheet in the Excel file, identifies cells containing questions, and calls the **write\_answer(question)** function from the second script to generate answers for each question.
6. It tokenizes and preprocesses the search term, removing stopwords.
7. It loads the most recent Excel file containing the QRA library with matching filename patterns from the config directory.
8. It calculates the similarity of each row in the loaded Excel file to the search term.
   1. The search term is tokenized (split into individual words) and preprocessed by converting it to lowercase and removing English stopwords. This results in a list of relevant search tokens.
   2. For each row in the Excel file, the combined text of the 'Question' and 'Supporting Answer' columns is tokenized and preprocessed in the same way as the search term, resulting in a list of relevant line tokens.
   3. The similarity score between the search term and the row is calculated by counting the number of search tokens that appear in the line tokens. In other words, the similarity score is the number of common tokens between the search term and the combined text of the 'Question' and 'Supporting Answer' columns in each row.
   4. For each row, a dictionary is created with the row data and its calculated similarity score. These dictionaries are stored in a list called matches.
   5. The matches list is sorted in descending order by the similarity score.
9. It selects the top 6 most similar answers, creates a list of dictionaries with the search term, top 10 matches, and messages array, and sends this data to the OpenAI API for generating a response.
   1. OpenAI receives the creativity and length parameters, question text and some coaching instructions.
   2. The coaching contains a "system" role and provides an instruction to the language model on how to generate answers. It tells the model to assume the role of "Rimini Street" and respond to customer questions without using conversational lead-ins or repeating the question.
   3. The message body contains the search term (the question) that needs an answer. It also mentions that the response should be drawn from the provided example questions and answers about Rimini Street.
   4. The second part of the message body has the "user" role and contains the top 10 most similar matches (answers) found in the Excel file as newline-separated text. The language model is expected to draw inspiration from these matches while generating the response.
10. The API response (answer) is stored in a CSV file, and the function returns the generated answer.
11. The questions, answers, and cell locations are stored in the Excel DataFrame.
12. After processing all questions, the Excel DataFrame is written to a new sheet in the Excel file called Roboface QRA, and a completion message is printed.
13. The data.csv file records audit information on precisely how the answer was created.
14. The program closes after the Excel completes processing.