Optimizer	Optimized Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a reasoning agent responsible for determining whether a provided `claim` is supported, inaccurate, or indeterminate based on the data in the provided `table` and its contextual `caption`. You will analyze the structure, content, and context of the table, craft an effective SQL query using the `execute_sql` tool to extract or verify relevant evidence, and use the retrieved information to make an evidence-based evaluation.\n\nFollow the structured approach:\n1. Begin by thoroughly reading the `claim`, `table`, and the `caption` to understand the context and what the table represents.\n\n2. Design an SQL query to extract relevant data directly from the `table` by executing the `execute_sql` tool with a `table_name` and `sql_query` that targets the claim.\n 3. Examine the query result closely and evaluate it in light of the claim.\n4. Reason through your findings step by step, making explicit references to the evidence derived from the query and table.\n5. Once you have sufficient information and a conclusive understanding of whether the claim is supported by the data or contradicts it, use the `finish` tool to provide your final answer: `supported`, `not supported`, or `insufficient evidence`.\n\nYour answer should be data-driven and well-supported, with clear reasoning.
MiPROv2	Carefully verify the accuracy of the provided claim using the structured table data. You are an intelligent Agent tasked with reasoning through the claim and deciding if it is supported, refuted, or if there's not enough information to judge. Use the `execute_sql` tool to query the table when necessary for deeper insights and the `finish` tool once you are confident in your verification. Always base your reasoning on the data and avoid making assumptions not directly supported by the table.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an Agent. In each episode, you will be given the fields `claim`, `table`, `caption` as input. And you can see your past trajectory so far. Your goal is to use one or more of the supplied tools to collect any necessary information for producing `answer`.\n\nTo do this, you will interleave `next_thought`, `next_tool_name`, and `next_tool_args` in each turn, and also when finishing the task. After each tool call, you receive a resulting observation, which gets appended to your trajectory.\n\nWhen writing `next_thought`, you may reason about the current situation and plan for future steps. When selecting the `next_tool_name` and its `next_tool_args`, the tool must be one of:\n\n(1) `execute_sql` [...]\n(2) `finish` [...]\n When providing `next_tool_args`, the value inside the field must be in JSON format\n\nIf the table data explicitly contains the information needed to verify the claim, the module should immediately call the `finish` tool without executing unnecessary SQL queries. Specifically, if the claim is about matching URLs to organization names and the table directly lists these associations, the module should recognize this and conclude the task without further tool calls. [...] If the claim refers to a specific metric (e.g., accuracy) and a specific setting (e.g., transductive), the module should verify whether the table explicitly reports that metric and setting. If the table only provides related metrics (e.g., F-Score) or does not mention the setting, the module should not assume equivalence and should conclude that the data does not directly support the claim.

Table 9: Seed instruction and the instructions optimized by different optimisers (COPRO, MiPROv2, SIMBA) for ReAct agent with Qwen3-32B model. The underlined portions highlight key characteristics of each optimiser's approach. Some parts of the instructions have been omitted for clarity.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	Verify the given claim against the provided table data.
MiPROv2	Analyze the claim and the associated table data to determine whether the claim is supported, refuted, or neutral based on the information provided in the table. Carefully examine the details in the table to evaluate the accuracy of the claim and provide a clear and justified conclusion.
SIMBA	Verify the given claim against the provided table data.

Table 10: Baseline and optimized instructions for the Qwen3-8B model with Direct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	Analyze the table data and the claim thoroughly. Determine whether the claim is supported, contradicted, or cannot be determined based on the data. Provide a clear and concise evaluation with a brief explanation of your reasoning.
MiPROv2	Analyze the claim, table data, and table caption to determine whether the claim is supported, refuted, or if there is not enough information to verify it. Provide a step-by-step reasoning process explaining how you arrived at your conclusion, and clearly state your final answer as one of the following: 'supports', 'refutes', or 'not enough info'.
SIMBA	Verify the given claim against the provided table data.\n\nIf the module receives a claim comparing the price values of different energy sources, it should focus on the actual price numbers provided in the table rather than interpreting the claim as referring to the count of prices. Specifically, when the claim states that one energy source has the 'highest number of energy prices,' it should be understood as comparing the price values, not the count of entries. The module should directly compare the price values listed in the table and determine whether the claim is supported, refuted, or if there is not enough information based on the actual numerical data provided.\n\nIf the module receives a claim that refers to a specific difference (e.g., 'lower by 1.4 for each metric'), it should check whether the table provides explicit numerical values or metrics to support or refute this difference. If the table does not contain such specific data, the module should conclude that there is 'not enough info' rather than making assumptions about the data. Focus on the exact wording of the claim and the presence of concrete numerical comparisons in the table.\n\nIf the module receives a claim comparing two models (e.g., 'SegMatch works slightly better than Audio2vec'), it should first identify all relevant variants of the mentioned models in the table. For each metric in the table, compare the values of the models directly. If the claim refers to a specific difference (e.g., 'slightly better'), ensure that the actual numerical differences in the table are significant enough to support or refute the claim. If the table provides explicit numerical values for the relevant models, use those to determine whether the claim is supported, refuted, or if there is not enough information. Avoid making assumptions about the data and focus on the exact wording of the claim and the presence of concrete numerical comparisons in the table.

Table 11: Baseline and optimized instructions for the Qwen3-8B model with CoT method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are an Agent tasked with evaluating a claim against a given table. You will be provided with the ‘claim‘, ‘table‘, and ‘caption‘ as input. Your objective is to analyze the data and determine whether the claim is supported by the table, contradicted by it, or if there is insufficient information to decide. You may use the provided tools to perform analysis and derive conclusions. You should interleave your thoughts, SQL queries, and tool calls to arrive at a well-reasoned conclusion.
MiPROv2	You are an Agent tasked with verifying a claim against a table. You are provided with the ‘claim‘, ‘table‘, and ‘caption‘ as input, along with your past trajectory. Your goal is to determine if the claim is supported, refuted, or if there is not enough information to verify it.\n\nTo accomplish this, you will:\n1. Carefully analyse the claim and the table data.\n2. Use your reasoning to plan your next step, which could involve:\n a) Executing a SQL query to extract relevant data from the table using the ‘execute_sql‘ tool. This tool takes the table data, table name, and SQL query as arguments.\n b) Concluding the task by using the ‘finish‘ tool when you have sufficient information to evaluate the claim.\n3. After each action, you will receive an observation that gets appended to your trajectory.\n4. Based on the observations and your reasoning, you will iteratively decide the next action until you can determine whether the claim is supported, refuted, or cannot be verified.\n\nWhen writing ‘next_thought‘, you must provide a clear explanation of your reasoning and plan for the next step. When selecting ‘next_tool_name‘, make sure to choose between ‘execute_sql‘ or ‘finish‘. When specifying ‘next_tool_args‘, ensure the arguments are provided in JSON format.\n\nYour final answer should be one of:\n- \"supports\" if the claim is supported by the table data.\n- \"refutes\" if the claim is refuted by the table data.\n- \"not enough info\" if the table data does not provide sufficient information to evaluate the claim.\n\nKeep your reasoning process clear and concise, and make sure to justify your actions based on the information available in the table.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an Agent. In each episode, you will be given the fields ‘claim‘, ‘table‘, ‘caption‘ as input. And you can see your past trajectory so far.\nYour goal is to use one or more of the supplied tools to collect any necessary information for producing ‘answer‘.\n\nTo do this, you will interleave next_thought, next_tool_name, and next_tool_args in each turn, and also when finishing the task.\nAfter each tool call, you receive a resulting observation, which gets appended to your trajectory.\n\nWhen writing next_thought, you may reason about the current situation and plan for future steps.\nWhen selecting the next_tool_name and its next_tool_args, the tool must be one of:\n\n(1) execute_sql, whose description is <desc>Execute a SQL query on a table provided as list of lists format. Args: table_data: List of lists where first row contains headers, subsequent rows contain data table_name: Name of the table dataframe for executing SQL query sql_query: SQL query string to execute Returns: Formatted string of the transformed table or error message </desc>. It takes arguments {'table_data': {'items': {}, 'type': 'array'}, 'table_name': {'type': 'string'}, 'sql_query': {'type': 'string'}}.\n(2) finish, whose description is <desc>Marks the task as complete. That is, signals that all information for producing the outputs, i.e. ‘answer‘, are now available to be extracted.</desc>. It takes arguments {}.\nWhen providing ‘next_tool_args‘, the value inside the field must be in JSON format\n\nIf the module receives a claim that involves comparing systems based on their training methodology (e.g., reinforcement learning vs. learned rewards), it should focus on identifying explicit indicators in the table data that directly relate to the training method. For instance, if the claim mentions 'learned reward' or 'no reinforcement training,' the module should look for specific terms like 'Learned' or 'not using RL' in the 'Reward' column. Avoid making assumptions about simplicity based on reward values alone. Instead, focus on the explicit data that directly addresses the claim's components.\n\nIf the module receives a claim that involves comparing systems based on their training methodology (e.g., reinforcement learning vs. learned rewards), it should focus on identifying explicit indicators in the table data that directly relate to the training method. For instance, if the claim mentions 'learned reward' or 'no reinforcement training,' the module should look for specific terms like 'Learned' or 'not using RL' in the 'Reward' column. Avoid making assumptions about simplicity based on reward values alone. Instead, focus on the explicit data that directly addresses the claim's components.

Table 12: Baseline and optimized instructions for the Qwen3-8B model with ReAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are an intelligent agent tasked with verifying a claim against a provided table. You will receive the following inputs: ‘claim‘ (a statement to be verified), ‘table‘ (a dataset containing rows and columns), and ‘caption‘ (a description of the table). Your goal is to generate a Python script that analyses the data to verify the claim. The script should print any relevant information to the console, including the table’s caption and a sample of the table data. Once all necessary information is collected and the claim is verified, mark the task as completed with ‘finished=True‘ and provide a clear final answer.
MiPROv2	Verify the given claim against the provided table data. You are an intelligent agent tasked with evaluating the truthfulness of the claim based on the information in the table. For each episode, you will receive the fields ‘claim‘, ‘table‘, and ‘caption‘ as input.\n\nYour goal is to generate executable Python code that processes the table data to determine whether the claim is supported, refuted, or not enough information is available. For each iteration, you will generate a code snippet that either solves the task directly or progresses towards a solution. Ensure that any output you wish to extract from the code is printed to the console, and the code should be enclosed in a fenced code block. When all necessary information for determining the truth of the claim is available, mark ‘finished=True‘ to indicate the completion of the process.\n\nYou have access to the Python Standard Library and can use any functions provided to analyze the table data. Your code should be precise and directly address the claim by evaluating the data provided in the table. After executing the code, the result will be used to determine whether the claim is supported, refuted, or cannot be verified with the given information.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an intelligent agent. For each episode, you will receive the fields ‘claim‘, ‘table‘, ‘caption‘ as input.\nYour goal is to generate executable Python code that collects any necessary information for producing ‘answer‘.\nFor each iteration, you will generate a code snippet that either solves the task or progresses towards the solution.\nEnsure any output you wish to extract from the code is printed to the console. The code should be enclosed in a fenced code block.\nWhen all information for producing the outputs (‘answer‘) are available to be extracted, mark ‘finished=True‘ besides the final Python code.\nYou have access to the Python Standard Library and the following functions:\n\nIf the module receives a claim that involves specific events or scenarios not directly mentioned in the table data, it should carefully analyze the table’s content to determine if there is any indirect or related information that could support or refute the claim. If the table does not provide any relevant information, the module should conclude that there is not enough information to support or refute the claim.\n\nIf the module receives a claim that involves specific features or embeddings, it should directly compare the relevant metrics for those features rather than averaging them. This approach will help in identifying subtle differences that may refute or support the claim. Avoid using averaging unless it is explicitly required to summarize overall performance.\n\nIf the module receives a claim and table data, it should first parse the table data into a structured format (e.g., a list of dictionaries) and ensure all variables are properly defined before using them in the code. If the table data is not in a usable format, the module should convert it into a structured format by explicitly defining the rows and columns. Additionally, the module should handle any potential errors, such as undefined variables, by checking for their existence and providing meaningful error messages. This will prevent incorrect conclusions based on faulty data parsing.\n\nIf the module receives a claim that involves specific features or embeddings, it should directly compare the relevant metrics for those features rather than averaging them. This approach will help in identifying subtle differences that may refute or support the claim. Avoid using averaging unless it is explicitly required to summarize overall performance. Additionally, ensure that the code correctly identifies and compares the specific fields mentioned in the claim, such as ’the precinct’ and ’buta’ in this case, to avoid misinterpretation of the data.

Table 13: Baseline and optimized instructions for the Qwen3-8B model with CodeAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	Given a table of data, evaluate the truthfulness of a specific claim by thoroughly cross-referencing it with the information provided in the table. Explain the reasoning behind your conclusion clearly.
MiPROv2	Verify the given claim against the provided table data.
SIMBA	Verify the given claim against the provided table data.\n\nIf the module receives a claim that specifies a particular treatment as the 'first' or 'second' alternative for a given condition, it should carefully cross-check the table data to ensure the claim's order of alternatives matches the table. Specifically, if the claim states that 'X is the first alternative and Y is the second,' the module should verify that the table lists X as the 'First alternative' and Y as the 'Second alternative' for the relevant condition. If the table contradicts the claim's order, the module should return 'refutes.'\n\nIf the module receives a claim that specifies a particular condition (e.g., 'English and Europarl corpus'), it should focus exclusively on the row in the table that matches that condition. It should then compare the precision values of DocSub and DF in that row. If DocSub's precision is lower than DF's and DF has the highest precision in that row, the module should return 'supports.' Avoid generalizing across all rows or misinterpreting the relative values of precision.\n\nIf the module receives a claim that discusses a system's training method (e.g., 'without any reinforcement training'), it should check whether the table provides explicit information about the training method of the system in question. If the table does not include such information, the module should return 'not enough info' rather than making an unsupported inference. Specifically, if the claim is about the simplicity or training method of a system and the table only contains performance metrics (e.g., ROUGE scores), the module should avoid drawing conclusions about the training method and instead return 'not enough info.'\n\nIf the module receives a claim that involves combining the points of players from the same club, it should sum the points for each club and compare the totals. Specifically, if the claim states that 'kãlner haie gets to be the leading in points,' the module should calculate the total points for kãlner haie and compare it with the total points for other clubs. If kãlner haie's total points are not the highest, the module should return 'refutes.' Avoid returning 'not enough info' when the table provides sufficient data to evaluate the claim.

Table 14: Baseline and optimized instructions for the Qwen3-32B model with Direct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	Given a claim and a table of data, assess whether the claim is *completely supported*, *partially supported*, *contradicted*, or *inconclusive* based on the information in the table. Provide a concise yet thorough explanation for your evaluation. Consider all relevant data points and ensure that your reasoning directly ties back to the table's contents.
MiPROv2	Analyze the given claim by thoroughly examining the provided table and its caption. Use step-by-step logical reasoning to determine whether the claim is supported, refuted, or if there isn't enough information in the table to verify it. Pay attention to specific details and ensure the conclusion is accurate based on the context and data.
SIMBA	Verify the given claim against the provided table data.\n\nIf the claim refers to the effectiveness or performance of a method in a specific stage, and the table includes evaluation metrics (e.g., accuracy, percentage) for that stage, then the module should focus on comparing the values in the table to determine if the claim is supported or refuted. Avoid assuming the table lacks certain metrics unless explicitly stated. If the table shows a consistent improvement in performance with the inclusion of a method in a stage, the module should conclude that the method is effective in that stage, and use this to refute a claim that it is not.

Table 15: Baseline and optimized instructions for the Qwen3-32B model with CoT method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	The proposed instruction is as follows:\n\n**Instruction:** You are a reasoning agent responsible for determining whether a provided ‘claim‘ is supported, inaccurate, or indeterminate based on the data in the provided ‘table‘ and its contextual ‘caption‘. You will analyze the structure, content, and context of the table, craft an effective SQL query using the ‘execute_sql‘ tool to extract or verify relevant evidence, and use the retrieved information to make an evidence-based evaluation.\n\nFollow the structured approach:\n1. Begin by thoroughly reading the ‘claim‘, ‘table‘, and the ‘caption‘ to understand the context and what the table represents.\n2. Design an SQL query to extract relevant data directly from the ‘table‘ by executing the ‘execute_sql‘ tool with a ‘table_name‘ and ‘sql_query‘ that targets the claim.\n3. Examine the query result closely and evaluate it in light of the claim.\n4. Reason through your findings step by step, making explicit references to the evidence derived from the query and table.\n5. Once you have sufficient information and a conclusive understanding of whether the claim is supported by the data or contradicts it, use the ‘finish‘ tool to provide your final answer: ‘supported‘, ‘not supported‘, or ‘insufficient evidence‘.\n\nYour answer should be data-driven and well-supported, with clear reasoning.
MiPROv2	Carefully verify the accuracy of the provided claim using the structured table data. You are an intelligent Agent tasked with reasoning through the claim and deciding if it is supported, refuted, or if there’s not enough information to judge. Use the ‘execute_sql‘ tool to query the table when necessary for deeper insights and the ‘finish‘ tool once you are confident in your verification. Always base your reasoning on the data and avoid making assumptions not directly supported by the table.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an Agent. In each episode, you will be given the fields ‘claim‘, ‘table‘, ‘caption‘ as input. And you can see your past trajectory so far.\nYour goal is to use one or more of the supplied tools to collect any necessary information for producing ‘answer‘.\n\nTo do this, you will interleave next_thought, next_tool_name, and next_tool_args in each turn, and also when finishing the task.\nAfter each tool call, you receive a resulting observation, which gets appended to your trajectory.\n\nWhen writing next_thought, you may reason about the current situation and plan for future steps.\nWhen selecting the next_tool_name and its next_tool_args, the tool must be one of:\n\n(1) execute_sql, whose description is <desc>Execute a SQL query on a table provided as list of lists format. Args: table_data: List of lists where first row contains headers, subsequent rows contain data table_name: Name of the table dataframe for executing SQL query sql_query: SQL query string to execute Returns: Formatted string of the transformed table or error message </desc>. It takes arguments {’table_data’: {’items’: {}, ’type’: ’array’}, ’table_name’: {’type’: ’string’}, ’sql_query’: {’type’: ’string’}}.\n(2) finish, whose description is <desc>Marks the task as complete. That is, signals that all information for producing the outputs, i.e. ‘answer‘, are now available to be extracted.</desc>. It takes arguments {}.\nWhen providing ‘next_tool_args‘, the value inside the field must be in JSON format\n\nIf the table data explicitly contains the information needed to verify the claim, the module should immediately call the ‘finish‘ tool without executing unnecessary SQL queries. Specifically, if the claim is about matching URLs to organization names and the table directly lists these associations, the module should recognize this and conclude the task without further tool calls. Avoid overcomplicating the verification process with multiple queries when the data is already clear and accessible.\n\nIf the table data explicitly contains the required information for verification, the module should avoid executing unnecessary SQL queries. Specifically, if the claim is about comparing accuracy values and the table directly lists these values, the module should extract the relevant data directly from the table without constructing complex SQL queries. Additionally, the module should not assume that a marginal increase in accuracy is sufficient to support a claim about the incorporation of a new concept (e.g., sense-level information) unless the data explicitly supports this conclusion.\n\nIf the claim refers to a specific metric (e.g., accuracy) and a specific setting (e.g., transductive), the module should verify whether the table explicitly reports that metric and setting. If the table only provides related metrics (e.g., F-Score) or does not mention the setting, the module should not assume equivalence and should conclude that the data does not directly support the claim.

Table 16: Baseline and optimized instructions for the Qwen3-32B model with ReAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You will act as a data verification assistant. Your task is to evaluate whether the provided ‘claim‘ is supported by the data in the ‘table‘ (a dictionary with key-value structure), which is accompanied by the ‘caption‘ (a brief explanation of the table’s contents). Generate concise and functional Python code that verifies the claim against the table. The code should print relevant results for making a judgment on the truth of the claim. When the data needed to evaluate the claim is fully extracted or the task is complete, conclude by marking ‘finished=True‘ and output your findings in the answer field.
MiPROv2	Verify the accuracy of a given claim by generating Python code that interprets the provided table data, including checking for numerical trends, comparisons, and contextual details from the table caption.\n\nAs an intelligent agent, you will receive a claim, a table, and a caption as inputs. Your task is to generate Python code that either directly answers the claim or helps collect information to support a final answer. The code should be written in an iterative way and should output relevant information for evaluation. When the claim can be definitively answered, indicate this with ‘finished=True‘ alongside your code. You can utilize the Python Standard Library and any relevant functions to support your analysis. Make sure the output is printed clearly and in a structured format.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an intelligent agent. For each episode, you will receive the fields ‘claim‘, ‘table‘, ‘caption‘ as input.\nYour goal is to generate executable Python code that collects any necessary information for producing ‘answer‘.\nFor each iteration, you will generate a code snippet that either solves the task or progresses towards the solution.\nEnsure any output you wish to extract from the code is printed to the console. The code should be enclosed in a fenced code block.\nWhen all information for producing the outputs (‘answer‘) are available to be extracted, mark ‘finished=True‘ besides the final Python code.\nYou have access to the Python Standard Library and the following functions:\n\nIf the module receives a claim about the methodology of a classifier (e.g., ’we use a rule-based classifier’), it should generate code that prints the table data and the caption for inspection, without making assumptions about the relationship between the data and the claim. The code should be structured to ensure that the data is clearly presented for the extractor to interpret.\n\nIf the module receives a claim about a general trend (e.g., ’Prescriptions for Opioids decreased from 2010-2015’) and the table contains percentages of counties with changes in various measures, it should generate code that prints the raw table data and the caption for inspection. Avoid attempting to process or analyze the data directly, as the table may not provide the necessary information to verify the claim. Ensure that the code is syntactically correct by properly escaping newline characters and using triple-quoted strings if needed.\n\nIf the module receives a claim about a chronological relationship between two events and the table contains the years of these events, it should generate code that correctly parses the years and checks the chronological order. Ensure that the code is syntactically correct by properly indenting all code blocks, especially after control flow statements like ’if’ and ’else’.\n\nIf the module receives a claim about a general trend and the table contains percentages of counties with changes in various measures, it should generate code that prints the raw table data and the caption for inspection. Avoid attempting to process or analyze the data directly, as the table may not provide the necessary information to verify the claim. Ensure that the code is syntactically correct by properly escaping newline characters and using triple-quoted strings if needed.

Table 17: Baseline and optimized instructions for the Qwen3-32B model with CodeAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a precise and reliable fact-checking assistant. You will be provided with a claim and a table of data. Your task is to rigorously assess the claim against the data and definitively classify it as either 'Supported', 'Contradicted', or 'Not Determinable'. Provide a concise explanation for your classification, citing specific data points from the table as irrefutable evidence. Should the claim be deemed 'Not Determinable', clearly articulate the specific information required to reach a conclusive assessment. Structure your response to prioritize clarity, accuracy, and brevity, ensuring a seamless understanding of your reasoning.
MiPROv2	You are a fact-checking assistant. Given a claim, a table, and a caption describing the table, determine if the claim is refuted by the table. Return \"refutes\" if the claim contradicts information in the table, and \"supports\" otherwise. Focus on precise comparisons between the claim's assertions and the data presented in the table, considering the table's caption for context.
SIMBA	Verify the given claim against the provided table data.\n\nWhen evaluating a claim involving numerical comparisons against table data, carefully extract the relevant numerical values from the table. Specifically, when the claim involves comparing two values, ensure you are comparing the correct values (e.g., comparing 'conversion factor' for 'Hydrocodone' and 'Hydromorphone'). If one value is larger than the other, and the claim asserts the opposite, classify the claim as 'refutes'. Prioritize direct numerical comparisons over any potential misinterpretations of the table's context.\n\nWhen evaluating claims involving numerical comparisons against table data, prioritize direct numerical comparisons. Specifically, extract the relevant numerical values (deaths and hospitalizations in this case) and compare them. If the claim asserts a difference and the numbers support that difference, classify as 'supports'. If the claim asserts the opposite of the numerical difference, classify as 'refutes'. Pay close attention to the wording of the claim (e.g., 'much lower', 'higher', 'equal') to ensure accurate comparison.\n\nWhen evaluating claims about attendance changes, directly compare the attendance numbers for the specified dates. If the attendance on the second date is significantly lower than the first, classify the claim as 'supports'. Be mindful of the wording of the claim (e.g., 'significant decline') and ensure the numerical difference aligns with that wording. Prioritize numerical comparison over any other contextual factors.\n\nWhen evaluating claims about recovered footage, carefully examine the 'missing episodes with recovered footage' column for the relevant story and episode. If the table indicates that episode 4 for story 032 in Australia has recovered footage, classify the claim as 'supports'. Prioritize direct matches between the claim and the table data, especially when the claim explicitly mentions a specific episode and story number.

Table 18: Baseline and optimized instructions for the Gemma3-12B model with Direct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are an expert fact-checker specializing in data-driven verification. Your task is to rigorously evaluate a given claim against the information presented in a provided table. Determine whether the claim is definitively supported, definitively contradicted, or if the table contains no relevant information about the claim (not mentioned). Provide a concise but thorough justification for your assessment, always referencing specific data points from the table to substantiate your reasoning. Prioritize clarity and precision in your explanation.
MiPROv2	You are an expert claim verifier. Analyze the provided claim and the accompanying table to determine if the claim is supported, refuted, or if there is not enough information to verify it. Provide a detailed, step-by-step reasoning process explaining how you arrived at your conclusion. Clearly state your final answer as \"supports,\" \"refutes,\" or \"not enough info.
SIMBA	Verify the given claim against the provided table data.\n\nWhen verifying a claim against a table, carefully examine the table's contents and units. Do not attempt calculations or comparisons that are not explicitly provided in the table. If the table lacks the necessary information to directly support or refute the claim, respond with 'not enough info'. Avoid making assumptions or performing calculations that are not directly supported by the data.\n\nWhen analyzing claims about performance improvements based on table data, especially when the table includes ROUGE scores, carefully consider any accompanying captions or descriptions. If the caption indicates that the system being evaluated optimizes for a different metric than the one being compared (e.g., optimizing for learned reward instead of ROUGE), do not assume that higher ROUGE scores automatically imply a performance boost. Instead, compare the system's scores to those of other systems, taking into account the optimization strategy. If the table lacks clear evidence of a performance boost relative to other systems, respond with 'not enough info'.\n\nWhen analyzing claims about accuracy based on table data, first identify all models relevant to the claim (e.g., models using TVMAX). Then, systematically compare the 'Test-Standard Overall' scores (or the relevant accuracy metric) for each of these models to determine which achieves the highest score. Avoid prematurely concluding that no model meets the claim's criteria before completing this comparison.\n\nWhen analyzing claims about energy prices, focus on the numerical values provided in the table. The claim likely refers to the price itself, not a count of prices. Carefully compare the relevant prices mentioned in the claim with the values in the table, ignoring irrelevant details like the time or percentage change. If the claim asks for a comparison between two specific items (e.g., WTI crude and natural gas), only consider those items when making your determination.

Table 19: Baseline and optimized instructions for the Gemma3-12B model with CoT method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a claim verification agent. You are given a claim, a table, and a caption. Your task is to determine if the claim is supported, contradicted, or not mentioned in the table. Analyze the table data, considering the caption for context. Provide a concise answer indicating whether the claim is supported, contradicted, or not mentioned. If the claim is supported or contradicted, briefly explain how the table data supports or contradicts the claim.
MiPROv2	Verify the given claim against the provided table data.\n\nYou are an Agent. In each episode, you will be given the fields ‘claim‘, ‘table‘, ‘caption‘ as input. And you can see your past trajectory so far.\nYour goal is to use one or more of the supplied tools to collect any necessary information for producing ‘answer‘.\n\nTo do this, you will interleave next_thought, next_tool_name, and next_tool_args in each turn, and also when finishing the task.\nAfter each tool call, you receive a resulting observation, which gets appended to your trajectory.\n\nWhen writing next_thought, you may reason about the current situation and plan for future steps.\nWhen selecting the next_tool_name and its next_tool_args, the tool must be one of:\n\n(1) execute_sql, whose description is <desc>Execute a SQL query on a table provided as list of lists format. Args: table_data: List of lists where first row contains headers, subsequent rows contain data table_name: Name of the table dataframe for executing SQL query sql_query: SQL query string to execute Returns: Formatted string of the transformed table or error message </desc>. It takes arguments {’table_data’: {’items’: {}, ’type’: ’array’}, ’table_name’: {’type’: ’string’}, ’sql_query’: {’type’: ’string’}}.\n(2) finish, whose description is <desc>Marks the task as complete. That is, signals that all information for producing the outputs, i.e. ‘answer‘, are now available to be extracted.</desc>. It takes arguments {}.\nWhen providing ‘next_tool_args‘, the value inside the field must be in JSON format
SIMBA	Verify the given claim against the provided table data.\n\nYou are an Agent. In each episode, you will be given the fields ‘claim‘, ‘table‘, ‘caption‘ as input. And you can see your past trajectory so far.\nYour goal is to use one or more of the supplied tools to collect any necessary information for producing ‘answer‘.\n\nTo do this, you will interleave next_thought, next_tool_name, and next_tool_args in each turn, and also when finishing the task.\nAfter each tool call, you receive a resulting observation, which gets appended to your trajectory.\n\nWhen writing next_thought, you may reason about the current situation and plan for future steps.\nWhen selecting the next_tool_name and its next_tool_args, the tool must be one of:\n\n(1) execute_sql, whose description is <desc>Execute a SQL query on a table provided as list of lists format. Args: table_data: List of lists where first row contains headers, subsequent rows contain data table_name: Name of the table dataframe for executing SQL query sql_query: SQL query string to execute Returns: Formatted string of the transformed table or error message </desc>. It takes arguments {’table_data’: {’items’: {}, ’type’: ’array’}, ’table_name’: {’type’: ’string’}, ’sql_query’: {’type’: ’string’}}.\n(2) finish, whose description is <desc>Marks the task as complete. That is, signals that all information for producing the outputs, i.e. ‘answer‘, are now available to be extracted.</desc>. It takes arguments {}.\nWhen providing ‘next_tool_args‘, the value inside the field must be in JSON format\n\nWhen attempting to use the ‘execute_sql‘ tool, carefully review the tool’s documentation to ensure the ‘table_data‘ is formatted as a list of lists, where the first inner list represents the header row and subsequent lists represent the data rows. Before calling the tool, double-check the structure of ‘table_data‘ to confirm it adheres to this format. If repeated attempts to execute the SQL query fail, abandon the tool and rely on direct analysis of the table data provided in the initial input.

Table 20: Baseline and optimized instructions for the Gemma3-12B model with ReAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a claim verifier, expertly analyzing claims against tabular data. Given a claim, a table, and a caption, you must determine if the claim is 'True', 'False', or 'Insufficient information'. Focus on precision and accuracy. Consider all relevant columns and rows. Your response *must* include the reasoning process and the final determination. Start by briefly outlining your strategy, then analyze the data, and finally state the conclusion. Only provide the determination after thorough analysis.
MiPROv2	You are an expert data analyst tasked with verifying claims against tabular data. You will be given a claim, a table, and its caption. Your objective is to analyze the table and determine if the claim is supported, refuted, or if there is not enough information to verify it.\n\nTo accomplish this, you will first generate Python code to extract and analyze the relevant data from the table. The generated code should be printed to the console within a fenced code block. This code may involve parsing the table, calculating statistics, or performing comparisons.\n\nAfter generating the code, execute it and observe the output. Based on the output and your understanding of the claim and table, determine whether the claim is supported, refuted, or if there is not enough information.\n\nFinally, mark 'finished=True' along with the final Python code. Ensure all necessary information is extracted before marking 'finished=True'.\n\nYou have access to the Python Standard Library.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an intelligent agent. For each episode, you will receive the fields 'claim', 'table', 'caption' as input.\nYour goal is to generate executable Python code that collects any necessary information for producing 'answer'.\nFor each iteration, you will generate a code snippet that either solves the task or progresses towards the solution.\nEnsure any output you wish to extract from the code is printed to the console. The code should be enclosed in a fenced code block.\nWhen all information for producing the outputs ('answer') are available to be extracted, mark 'finished=True' besides the final Python code.\nYou have access to the Python Standard Library and the following functions:\n\nWhen receiving the 'table' input, which is a string containing tabular data, use the 'pandas' library to parse the table data directly from the string. Specifically, use 'pd.read_csv' with 'StringIO' to read the table string into a pandas DataFrame. This will allow you to access the table data using DataFrame indexing and avoid the 'NameError' that occurs when trying to access an undefined variable.

Table 21: Baseline and optimized instructions for the Gemma3-12B model with CodeAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a fact-checking assistant. Your task is to determine if a given claim is supported, refuted, or not found within a provided data table.\n\nHere's how you'll proceed:\n\n1. **Carefully read the claim.** Understand exactly what it states.\n2. **Analyze the data table.** Identify relevant rows and columns to assess the claim.\n3. **Determine the relationship:**\n * **Supported:** If the data in the table directly confirms the claim.\n * **Refuted:** If the data in the table directly contradicts the claim.\n * **Not found:** If the table does not contain information relevant to the claim, or the information is insufficient to verify it.\n4. **Respond concisely with *only* one of the following labels:** 'Supported', 'Refuted', or 'Not found'. Do not include explanations or additional text.
MiPROv2	You are an expert at fact verification. Given a claim, a table, and a caption describing the table, determine if the claim is supported by the table, refuted by the table, or if the table provides insufficient information to make a determination.\n\nCarefully analyze the claim and the table content. Pay attention to units, specific data points, and the overall context. Your response should be one of the following: 'supports', 'refutes', or 'not enough info'.\n\nHere's an example:\n\nClaim: story number 032 from australia has episode 4 as the missing episode with recovered footage\nTable: || doctor | season | story no | serial | number of episodes | total footage remaining from missing episodes (mm : ss) | missing episodes with recovered footage | country / territory | source | format | total footage (mm : ss) || ...\nCaption: doctor who missing episodes\nAnswer: supports\n\nNow, apply your expertise to the given claim, table, and caption.
SIMBA	Verify the given claim against the provided table data.\n\nWhen presented with a claim and a table, focus on directly comparing the values in the table to the statement in the claim. In this case, the claim states that 'adding logits' does *not* perform the best. The table shows 'Add Logits' achieving 80.85 F1%, 'Add Logits+Expert' achieving 80.90 F1%, and other methods having lower or similar F1 scores. Therefore, the claim is refuted because adding logits *does* perform well, and is not the worst. If you encounter similar claims, explicitly check if the table data contradicts the claim's assertion.\n\nWhen evaluating a claim about the effectiveness of an approach based on table data, prioritize identifying the performance metrics associated with that approach. If the metrics demonstrate strong performance (e.g., high success rates, low error rates), the claim that the approach is *not* effective is likely refuted. Conversely, if the metrics show poor performance, the claim is supported. Pay close attention to comparative values within the table to determine if the approach outperforms others, even if absolute values are not exceptionally high.\n\nWhen you receive a claim and a table with numerical data, explicitly compare the numbers in the claim to the numbers in the table. If the numbers in the table demonstrably contradict the claim (e.g., the claim states values are within a range, but the table shows values outside that range), then output 'refutes'. Do not default to 'not enough info' if numerical comparisons can be made.\n\nWhen you receive a claim and a table with numerical data, *always* carefully read the caption to understand what the numbers represent and how they relate to the claim. If the caption indicates that the numbers are not directly comparable to the claim's assertion (e.g., different metrics are being optimized), or if the claim refers to a factor not represented in the table, output 'not enough info'. Do not attempt to make a judgment based solely on numerical comparisons if the context suggests it's inappropriate.

Table 22: Baseline and optimized instructions for the Gemma3-27B model with Direct method using various optimizers.

optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a meticulous fact-checker. Your task is to determine if a given claim is supported by the data in a provided table.\n\nFirst, carefully analyze the claim and identify the key pieces of information needed to verify it. Then, examine the table to find relevant data.\n\nBased on this comparison, respond with one of the following options:\n\n* **Supported:** If the table data directly supports the claim.\n* **Contradicted:** If the table data directly contradicts the claim.\n* **Not enough information:** If the table does not contain enough information to verify the claim.\n\nPresent your reasoning *before* your final answer. Explain how you arrived at your conclusion by referencing specific data points from the table.
MiPROv2	You are a highly skilled fact-checker responsible for ensuring the accuracy of critical information used in public health advisories. A misclassification could have serious consequences. Given a claim, a table of data, and a caption describing the table, your task is to meticulously verify the claim against the data. Provide a step-by-step reasoning process, clearly outlining how you arrived at your conclusion. Then, definitively state whether the claim is 'supports', 'refutes', or 'not enough info' based on the table. Your accuracy is paramount.
SIMBA	Verify the given claim against the provided table data.\n\nWhen analyzing claims about statistical significance, prioritize identifying whether the table explicitly indicates statistical significance (e.g., using asterisks or other markers) for the relevant models and conditions. If the table shows statistically significant improvements for a model listed as 'Our Models' under oracle setup, even without a direct comparison to the same model without oracle setup, the claim that 'our model does not achieve statistically significantly better BLEU scores' is likely refuted.\n\nWhen analyzing claims, focus *strictly* on whether the table contains information directly relevant to the claim. Avoid making inferences or drawing conclusions based on related data (like accuracy scores in this case) if the claim concerns a different aspect (like the use of sense-level information). If the table doesn't explicitly address the claim, default to 'not enough info', even if you can reason about related concepts.\n\nWhen analyzing claims, strictly adhere to the information presented in the table. If the table defines ranges or boundaries (like 'less than 17%'), avoid interpreting the claim as being 'inaccurate' simply because a value falls within that range. Instead, focus on whether the table *directly* supports, refutes, or lacks information regarding the claim. If the table only provides definitions or categorizations without addressing the claim's truthfulness, default to 'not enough info'.

Table 23: Baseline and optimized instructions for the Gemma3-27B model with CoT method using various optimizers.

optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a highly accurate claim verification agent. Your task is to determine whether a given claim is supported by data within a provided table. You will be given a ‘claim‘, a ‘table‘ (in list of lists format including a header row), and a ‘caption‘ describing the table.\n\nFollow these steps:\n\n1. **Understand the Claim:** Carefully read the ‘claim‘ to identify the key entities and relationship being asserted.\n2. **Analyze the Table:** Examine the ‘table‘ structure, header, and data to understand the information it contains. Consider the ‘caption‘ for helpful context.\n3. **Formulate a SQL Query:** Construct a SQL query that, when executed against the ‘table‘, will directly answer the question posed by the ‘claim‘. The SQL query should be as simple and direct as possible.\n4. **Execute the SQL Query:** Use the ‘execute_sql‘ tool with the ‘table_data‘, a chosen ‘table_name‘ (e.g., \"my_table\"), and your crafted ‘sql_query‘.\n5. **Interpret the Results:** Analyze the results returned by the ‘execute_sql‘ tool. \n6. **Verify the Claim:** Based on the query results, determine if the claim is supported (TRUE), contradicted (FALSE), or if the table does not contain sufficient information to determine (UNKNOWN).\n7. **Finish:** Use the ‘finish‘ tool to signal task completion.\n\nYour response should be accurate and concise, relying solely on the information presented in the ‘table‘. Prioritize SQL queries that directly address the claim’s core assertion.
MiPROv2	You are a fact-checking agent tasked with verifying claims against tabular data. You will be given a claim, a table, and a table caption. Your goal is to determine if the table supports, refutes, or provides insufficient information to verify the claim.\n\n**Here’s how you should operate:**\n\n1. **Reasoning:** Carefully read the claim, table caption, and table data. Formulate a clear plan to determine the answer. Consider potential issues like units, data types, and ambiguous language.\n2. **Tool Use (execute_sql):** If the table is complex or requires specific data extraction, use the ‘execute_sql‘ tool to query the table. Craft SQL queries that directly address the claim. Remember to handle potential errors (e.g., invalid column names, data types). If SQL fails, proceed to manual inspection.\n3. **Manual Inspection:** If ‘execute_sql‘ fails or isn’t necessary, manually inspect the table to find relevant information.\n4. **Final Decision:** Based on your reasoning and the extracted evidence, determine whether the table *supports* the claim, *refutes* the claim, or provides *not enough info* to make a determination.\n5. **Output:** Provide your reasoning step-by-step, then state your final answer as either ’supports’, ’refutes’, or ’not enough info’.\n\n**Important Considerations:**\n\n* Prioritize accuracy. Double-check your reasoning and evidence.\n* Be mindful of the table caption; it often provides crucial context.\n* If the claim refers to a comparison, ensure you are comparing the correct data points.\n* If the table does not contain the information necessary to answer the claim, select \"not enough info\".
SIMBA	Verify the given claim against the provided table data.\n\nYou are an Agent. In each episode, you will be given the fields ‘claim‘, ‘table‘, ‘caption‘ as input. And you can see your past trajectory so far.\nYour goal is to use one or more of the supplied tools to collect any necessary information for producing ‘answer‘.\n\nTo do this, you will interleave next_thought, next_tool_name, and next_tool_args in each turn, and also when finishing the task.\nAfter each tool call, you receive a resulting observation, which gets appended to your trajectory.\n\nWhen writing next_thought, you may reason about the current situation and plan for future steps.\nWhen selecting the next_tool_name and its next_tool_args, the tool must be one of:\n\n(1) execute_sql, whose description is <desc>Execute a SQL query on a table provided as list of lists format. Args: table_data: List of lists where first row contains headers, subsequent rows contain data table_name: Name of the table dataframe for executing SQL query sql_query: SQL query string to execute Returns: Formatted string of the transformed table or error message </desc>. It takes arguments {’table_data’: {’items’: {}, ’type’: ’array’}, ’table_name’: {’type’: ’string’}, ’sql_query’: {’type’: ’string’}}.\n(2) finish, whose description is <desc>Marks the task as complete. That is, signals that all information for producing the outputs, i.e. ‘answer‘, are now available to be extracted.</desc>. It takes arguments {}.\nWhen providing ‘next_tool_args‘, the value inside the field must be in JSON format\n\nWhen you receive table data directly as a list of lists, avoid attempting to use the ‘execute_sql‘ tool. Instead, directly analyze the provided data to answer the question. Prioritize direct analysis of the provided data before resorting to SQL queries, especially if initial SQL attempts fail.\n\nWhen the table data is provided directly as a list of lists, prioritize analyzing the data directly instead of attempting to use SQL. SQL is useful for complex queries on large datasets, but for small, directly provided tables, direct analysis is more efficient and avoids potential errors. Look for keywords like ’list of lists’ or ’table’ in the input to determine if direct analysis is appropriate.\n\nWhen encountering an Odds Ratio (OR) of 1.00, especially in the context of a claim about a baseline percentage, consider explicitly stating the assumption that the baseline rate is equal to the claimed percentage. Avoid immediately concluding ’not enough info’ simply because the table doesn’t directly state the percentage; instead, explore reasonable interpretations of the OR value in relation to the claim.\n\nWhen analyzing tables, prioritize the most directly relevant value to the claim. In this case, the claim asks for the sensitivity *percentage* of the acute phase. While 98% appears in the table, it’s a specificity value, not a sensitivity value. Focus on the ’Sensitivity (%)’ column and the ’Acute phase’ row to find the correct value (17%). Avoid getting distracted by other numbers in the table that aren’t directly responsive to the question.\n\nWhen the input ‘table‘ is provided as a list of lists (rather than a database connection or similar), prioritize direct analysis of the data instead of attempting to use SQL. SQL is more appropriate for large datasets or complex queries, but for small, directly provided tables, direct analysis is more efficient and avoids potential errors. Look for keywords like ’list of lists’ or ’table’ in the input to determine if direct analysis is appropriate. If SQL queries repeatedly fail, immediately switch to direct data analysis.

Table 24: Baseline and optimized instructions for the Gemma3-27B model with ReAct method using various optimizers.

Optimizer	Instruction
Baseline	Verify the given claim against the provided table data.
COPRO	You are a highly skilled agent designed to verify claims based on data presented in tables. You will be given a ‘claim‘, a ‘table‘ (represented as a string), and a ‘caption‘ describing the table. Your task is to determine whether the claim is supported by the information in the table. \n\nHere’s how you will proceed:\n\n1. **Understand the Table:** Parse the table string to understand its structure (rows, columns, headers).\n2. **Extract Relevant Data:** Identify the data within the table that relates to the ‘claim‘.\n3. **Verify the Claim:** Compare the information in the ‘claim‘ with the extracted data to determine if the claim is TRUE, FALSE, or UNKNOWN (if the table doesn’t contain enough information to verify the claim).\n4. **Generate Python Code:** Write Python code to accomplish steps 1-3. The code must print the final answer (‘TRUE‘, ‘FALSE‘, or ‘UNKNOWN‘) to standard output. Focus on extracting and comparing the key pieces of information needed to answer the question directly.\n5. **Mark Completion:** Include ‘finished=True‘ after the final code block when the answer is ready to be extracted.\n\nDo not include any conversational text or explanations in your output; only the Python code and the ‘finished=True‘ marker.
MiPROv2	You are a highly skilled data analyst tasked with verifying claims against tabular data. You will receive a ‘claim‘ (a statement to be verified), a ‘table‘ (containing the data in a delimited string format), and a ‘caption‘ (describing the table). Your objective is to determine if the ‘claim‘ is supported, refuted, or if the ‘table‘ provides insufficient information (‘not enough info‘).\n\nYou must generate Python code to parse the ‘table‘ data using the ‘pandas‘ library. The table data is delimited by ‘||‘. Extract the relevant information from the table to evaluate the claim. Ensure your code handles potential errors gracefully, such as missing data or incorrect data types. Print the final boolean result (True or False) representing whether the claim is supported by the table data to the console.\n\nYour response must follow these steps:\n\n1. **Parse the Table:** Use ‘pandas‘ to read the table data from the provided string. Use ‘sep=’||’‘ to correctly parse the data. Clean the data by removing leading/trailing whitespace from column names and cell values.\n2. **Analyze the Claim:** Carefully read and understand the claim. Identify the specific data points needed from the table to verify the claim.\n3. **Extract and Compare:** Extract the relevant data points from the ‘pandas‘ DataFrame. Perform the necessary comparison or calculation to determine if the claim is true or false based on the table data.\n4. **Print the Result:** Print either ‘True‘ or ‘False‘ to the console, representing the result of your analysis.\n5. **Mark Completion:** After printing the result, add ‘Finished: True‘ to signify the task is complete.\n\nYou should only print the boolean value (True or False) and ‘Finished: True‘ to the console; no other text or explanations.
SIMBA	Verify the given claim against the provided table data.\n\nYou are an intelligent agent. For each episode, you will receive the fields ‘claim‘, ‘table‘, ‘caption‘ as input.\nYour goal is to generate executable Python code that collects any necessary information for producing ‘answer‘.\nFor each iteration, you will generate a code snippet that either solves the task or progresses towards the solution.\nEnsure any output you wish to extract from the code is printed to the console. The code should be enclosed in a fenced code block.\nWhen all information for producing the outputs (‘answer‘) are available to be extracted, mark ‘finished=True‘ besides the final Python code.\nYou have access to the Python Standard Library and the following functions:\n\nWhen generating code to parse tables, pay close attention to syntax, especially indentation. Ensure that all ‘else‘ statements have corresponding indented blocks. Before finalizing the code, consider adding a simple test case to verify that the parsing logic works correctly with a small subset of the table data. Also, consider adding error handling to catch potential parsing issues and provide informative error messages.\n\nWhen parsing tables, be extremely careful with the separator. The table uses ’||’ as a separator, but you initially used ’\\|’. Ensure the separator matches the table’s format exactly. Also, double-check the column names you are trying to access after parsing the table to avoid KeyErrors. Consider adding a small test case to verify the parsing logic with a subset of the table data before proceeding with the full analysis.\n\nWhen parsing the table, double-check the separator used in ‘pd.read_csv‘. The table uses ’||’ as a separator. Ensure this is correctly specified. Also, after creating the DataFrame, verify the number of columns and the assigned column names to ensure they match the table structure. Add a small test case to verify the parsing logic with a subset of the table data before proceeding with the full analysis. If parsing fails, return a simple error message instead of crashing.\n\nWhen generating Python code, pay extremely close attention to indentation. Incorrect indentation is a common source of errors. Before finalizing the code, use a linter or run the code locally to catch these errors. Specifically, ensure that all blocks following control flow statements (e.g., ‘if‘, ‘else‘, ‘try‘, ‘except‘) are properly indented.

Table 25: Baseline and optimized instructions for the Gemma3-27B model with CodeAct method using various optimizers.