Raj Vegad

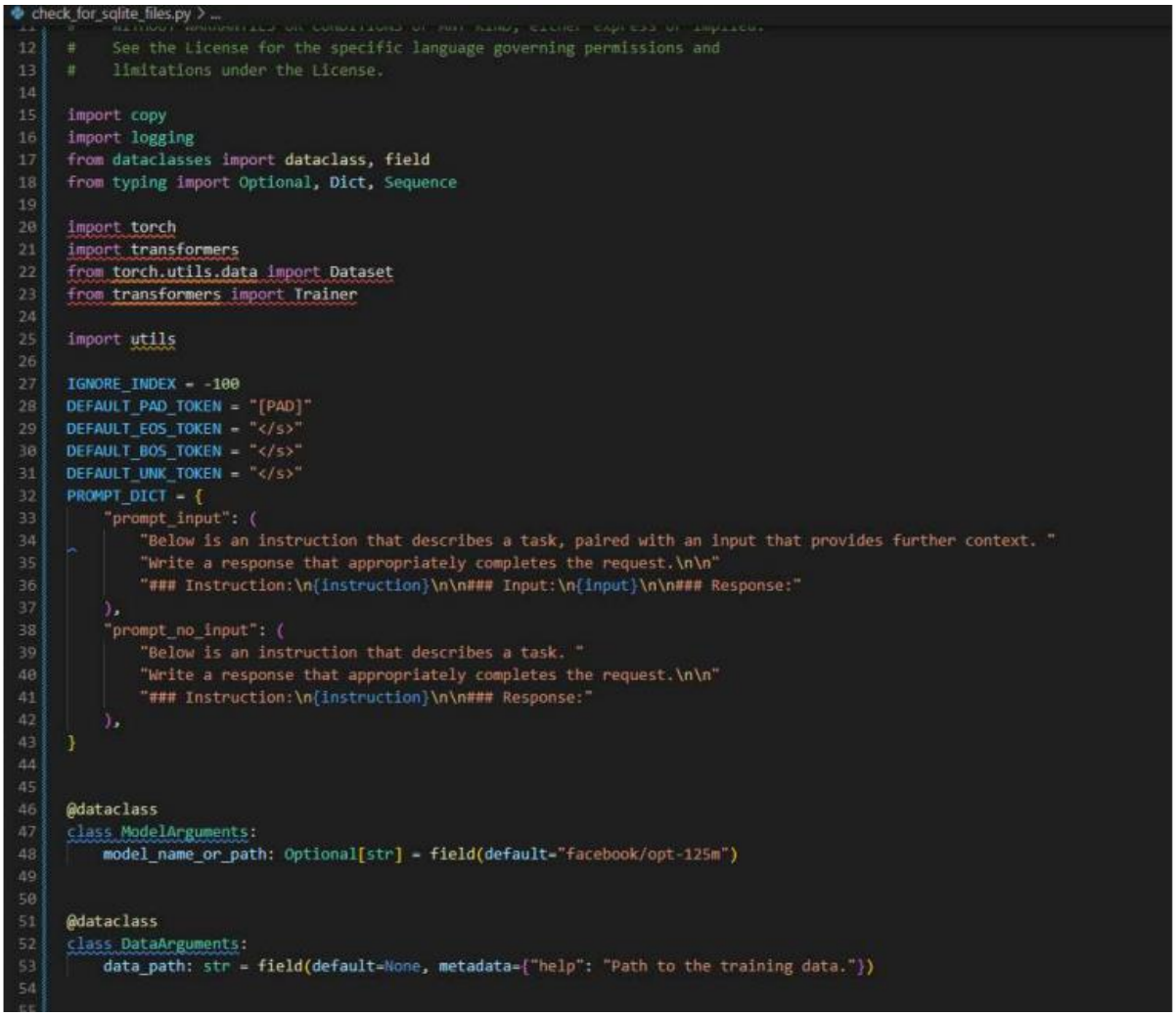
202001160

G-18

Lab-5

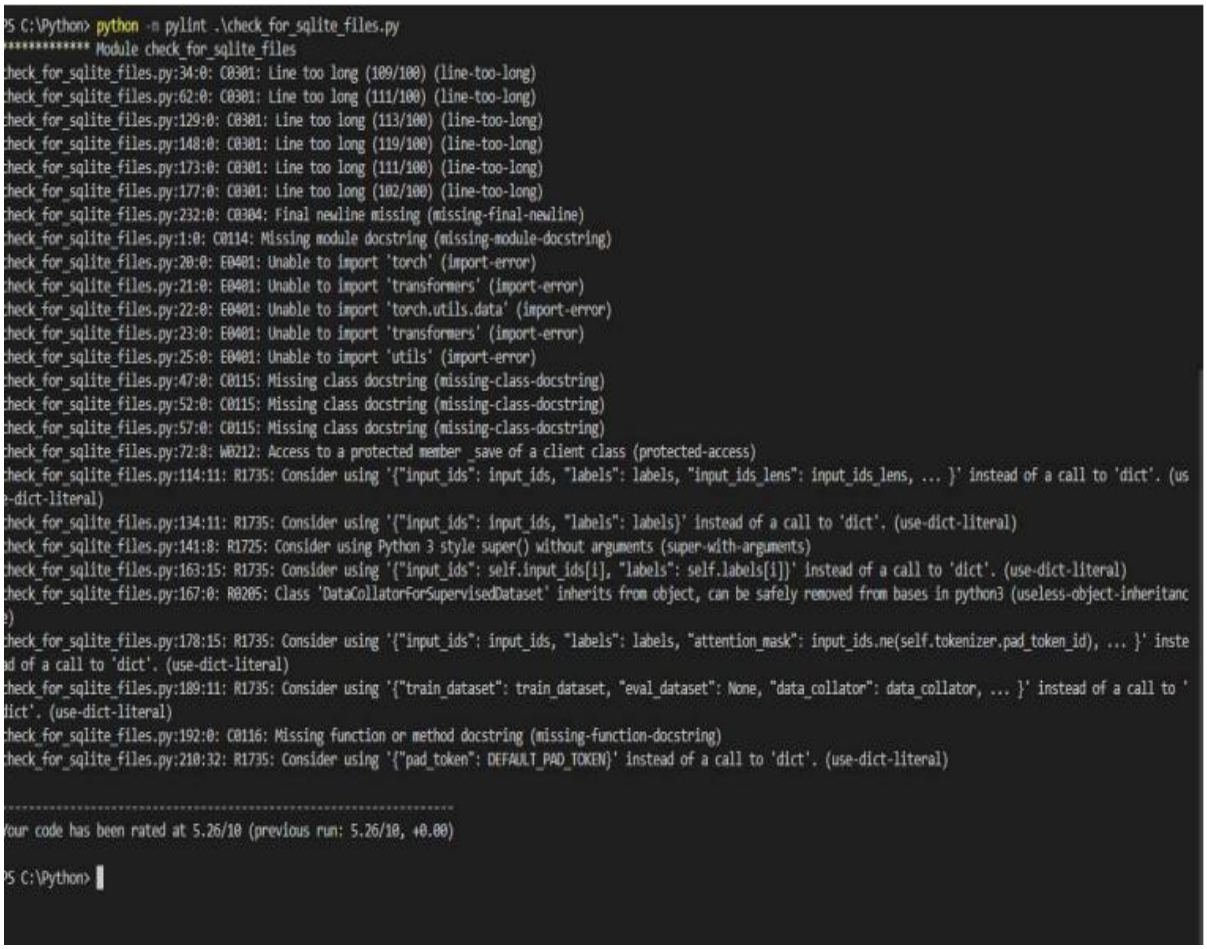
Software Engineering

Code:



```python
check_for_sqlite_files.py > ...
#    See the License for the specific language governing permissions and
#    limitations under the License.

import copy
import logging
from dataclasses import dataclass, field
from typing import Optional, Dict, Sequence

import torch
import transformers
from torch.utils.data import Dataset
from transformers import Trainer

import utils

IGNORE_INDEX = -100
DEFAULT_PAD_TOKEN = "[PAD]"
DEFAULT_EOS_TOKEN = "</s>"
DEFAULT_BOS_TOKEN = "</s>"
DEFAULT_UNK_TOKEN = "</s>"
PROMPT_DICT = {
    "prompt_input": (
        "Below is an instruction that describes a task, paired with an input that provides further context. "
        "Write a response that appropriately completes the request.\n\n"
        "### Instruction:\n{instruction}\n\n### Input:\n{input}\n\n### Response:"
    ),
    "prompt_no_input": (
        "Below is an instruction that describes a task. "
        "Write a response that appropriately completes the request.\n\n"
        "### Instruction:\n{instruction}\n\n### Response:"
    ),
}


@dataclass
class ModelArguments:
    model_name_or_path: Optional[str] = field(default="facebook/opt-125m")


@dataclass
class DataArguments:
    data_path: str = field(default=None, metadata={"help": "Path to the training data."})
```

Static analysis:



Code:

Chrome Dino Automater.py > ...

```python
import pyautogui  # pip install pyautogui
from PIL import Image, ImageGrab  # pip install pillow

# from numpy import asarray
import time


def hit(key):
    pyautogui.press(key)
    return


def isCollide(data):

    # for cactus
    for i in range(329, 425):
        for j in range(550, 650):
            if data[i, j] < 100:
                hit("up")
                return

    # Draw the rectangle for birds
    # for i in range(310, 425):
    #     for j in range(390, 550):
    #         if data[i, j] < 100:
    #             hit("down")
    #             return

    # return


if __name__ == "__main__":
    print("Hey.. Dino game about to start in 3 seconds")
    time.sleep(2)
    # hit('up')

    while True:
        image = ImageGrab.grab().convert("L")
        data = image.load()
        isCollide(data)

        # print(aarray(image))

        # Draw the rectangle for cactus
        # for i in range(315, 425):
        #     for j in range(550, 650):
        #         data[i, j] = 0
```

Static analysis:

```
S C:\Python> python -m pylint '.\Chrome Dino Automater.py'
********** Module Chrome Dino Automater
hrome Dino Automater.py:1:0: C0114: Missing module docstring (missing-module-docstring)
hrome Dino Automater.py:1:0: C0103: Module name "Chrome Dino Automater" doesn't conform to snake_case naming style (invalid-name)
hrome Dino Automater.py:1:0: E0401: Unable to import 'pyautogui' (import-error)
hrome Dino Automater.py:8:0: C0116: Missing function or method docstring (missing-function-docstring)
hrome Dino Automater.py:8:0: R1711: Useless return at end of function or method (useless-return)
hrome Dino Automater.py:13:0: C0116: Missing function or method docstring (missing-function-docstring)
hrome Dino Automater.py:13:0: C0103: Function name "isCollide" doesn't conform to snake_case naming style (invalid-name)
hrome Dino Automater.py:13:14: W0621: Redefining name 'data' from outer scope (line 39) (redefined-outer-name)
hrome Dino Automater.py:5:0: C0411: standard import "import time" should be placed before "import pyautogui" (wrong-import-order)
hrome Dino Automater.py:2:0: W0611: Unused Image imported from PIL (unused-import)

--------------------------------
our code has been rated at 2.63/10
```

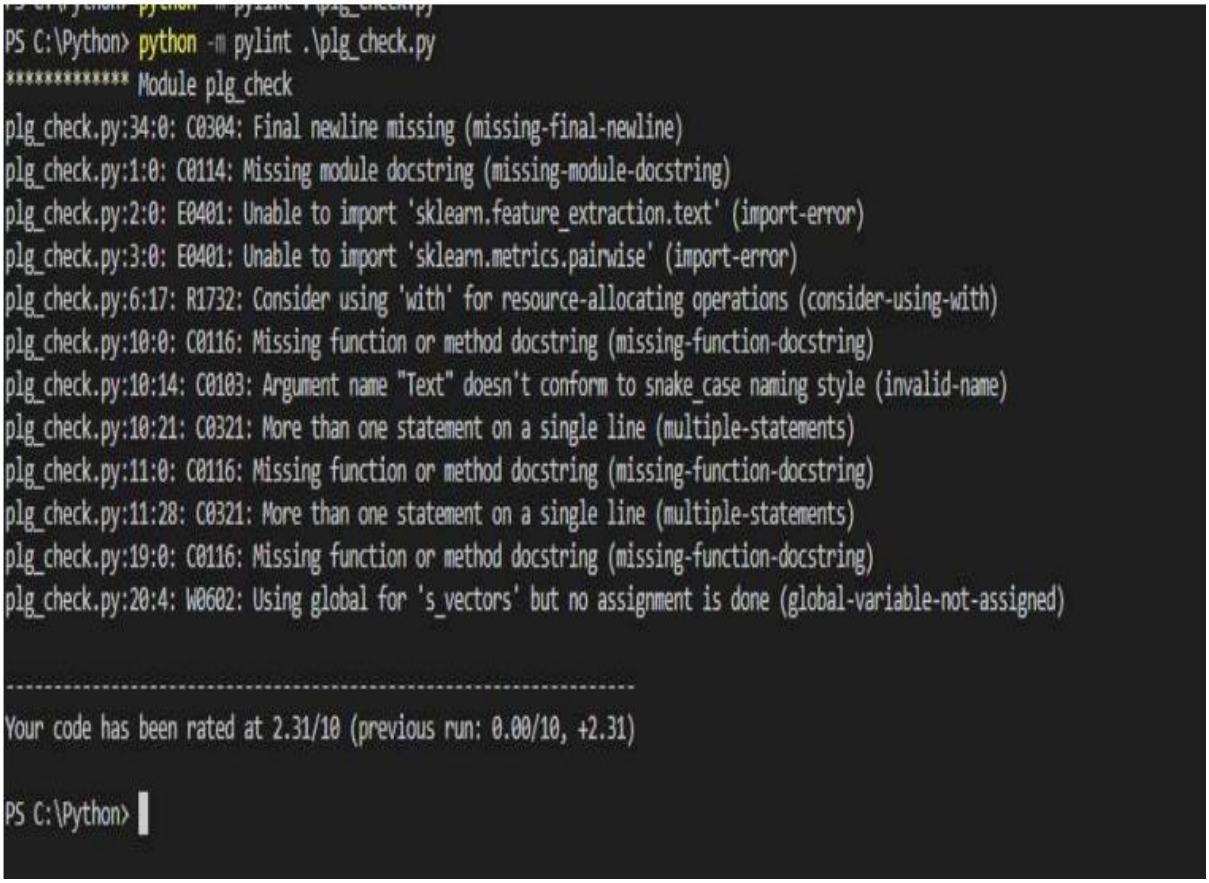
Code:

plg_check.py 5, U

plg_check.py > ...

```python
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

student_files = [doc for doc in os.listdir() if doc.endswith('.txt')]
student_notes = [open(_file, encoding='utf-8').read()
                 for _file in student_files]


def vectorize(Text): return TfidfVectorizer().fit_transform(Text).toarray()
def similarity(doc1, doc2): return cosine_similarity([doc1, doc2])


vectors = vectorize(student_notes)
s_vectors = list(zip(student_files, vectors))
plagiarism_results = set()


def check_plagiarism():
    global s_vectors
    for student_a, text_vector_a in s_vectors:
        new_vectors = s_vectors.copy()
        current_index = new_vectors.index((student_a, text_vector_a))
        del new_vectors[current_index]
        for student_b, text_vector_b in new_vectors:
            sim_score = similarity(text_vector_a, text_vector_b)[0][1]
            student_pair = sorted((student_a, student_b))
            score = (student_pair[0], student_pair[1], sim_score)
            plagiarism_results.add(score)
    return plagiarism_results


for data in check_plagiarism():
    print(data)
```

Static Analysis:

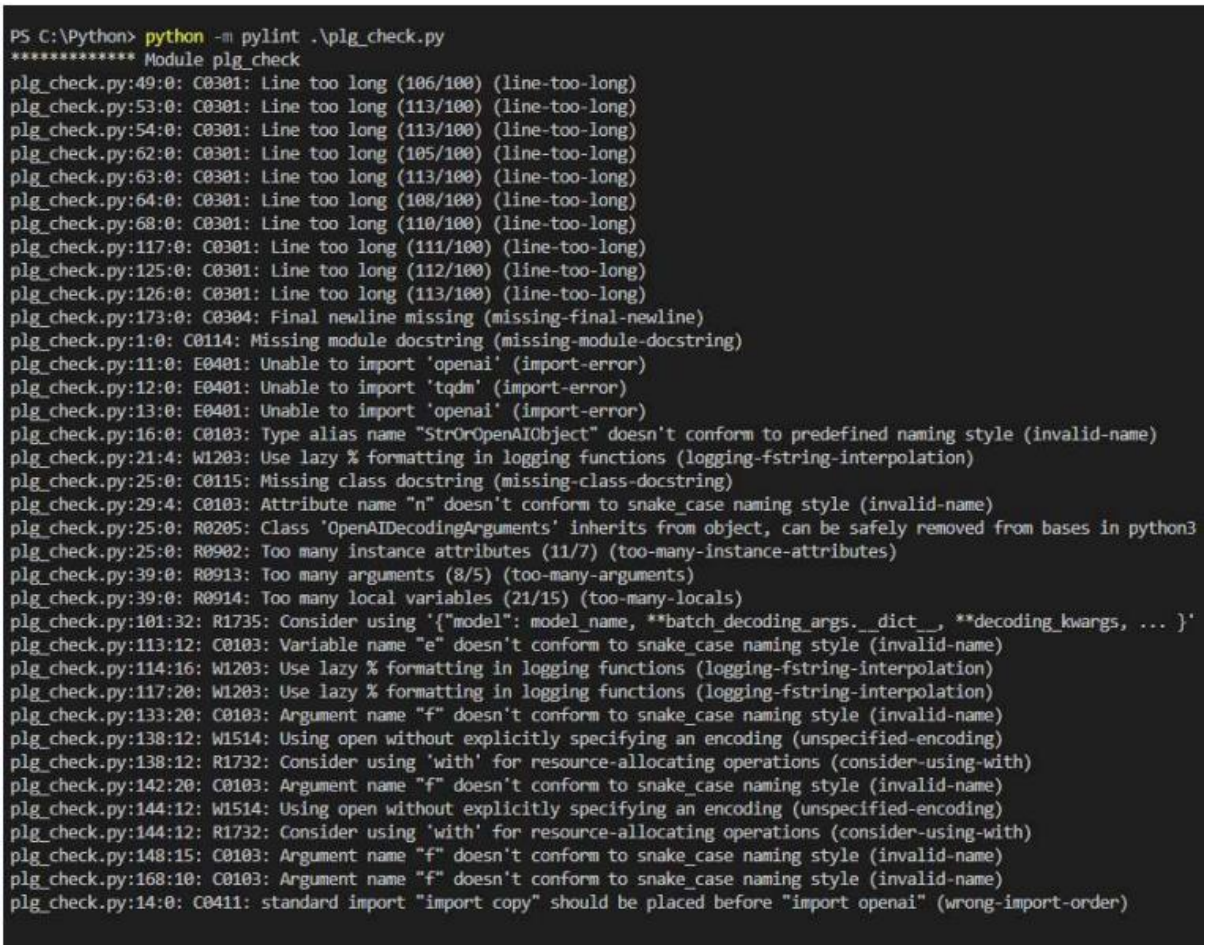

PS C:\Python> python -m pylint .\plg_check.py
********** Module plg_check
plg_check.py:34:0: C0304: Final newline missing (missing-final-newline)
plg_check.py:1:0: C0114: Missing module docstring (missing-module-docstring)
plg_check.py:2:0: E0401: Unable to import 'sklearn.feature_extraction.text' (import-error)
plg_check.py:3:0: E0401: Unable to import 'sklearn.metrics.pairwise' (import-error)
plg_check.py:6:17: R1732: Consider using 'with' for resource-allocating operations (consider-using-with)
plg_check.py:10:0: C0116: Missing function or method docstring (missing-function-docstring)
plg_check.py:10:14: C0103: Argument name "Text" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:10:21: C0321: More than one statement on a single line (multiple-statements)
plg_check.py:11:0: C0116: Missing function or method docstring (missing-function-docstring)
plg_check.py:11:28: C0321: More than one statement on a single line (multiple-statements)
plg_check.py:19:0: C0116: Missing function or method docstring (missing-function-docstring)
plg_check.py:20:4: W0602: Using global for 's_vectors' but no assignment is done (global-variable-not-assigned)

Your code has been rated at 2.31/10 (previous run: 0.00/10, +2.31)
PS C:\Python>

Code:

```python
import dataclasses
import logging
import math
import os
import io
import sys
import time
import json
from typing import Optional, Sequence, Union

import openai
import tqdm
from openai import openai_object
import copy

StrOrOpenAIObject = Union[str, openai_object.OpenAIObject]

openai_org = os.getenv("OPENAI_ORG")
if openai_org is not None:
    openai.organization = openai_org
    logging.warning(f"Switching to organization: {openai_org} for OAI API key.")


@dataclasses.dataclass
class OpenAIDecodingArguments(object):
    max_tokens: int = 1800
    temperature: float = 0.2
    top_p: float = 1.0
    n: int = 1
    stream: bool = False
    stop: Optional[Sequence[str]] = None
    presence_penalty: float = 0.0
    frequency_penalty: float = 0.0
    suffix: Optional[str] = None
    logprobs: Optional[int] = None
    echo: bool = False


def openai_completion(
    prompts: Union[str, Sequence[str], Sequence[dict[str, str]], dict[str, str]],
    decoding_args: OpenAIDecodingArguments,
    model_name="text-davinci-003",
    sleep_time=2,
    batch_size=1,
    max_instances=sys.maxsize,
    max_batches=sys.maxsize,
    return_text=False,
    **decoding_kwargs,
) -> Union[Union[StrOrOpenAIObject], Sequence[StrOrOpenAIObject], Sequence[Sequence[StrOrOpenAIObject]],]:
```

Static Analysis:

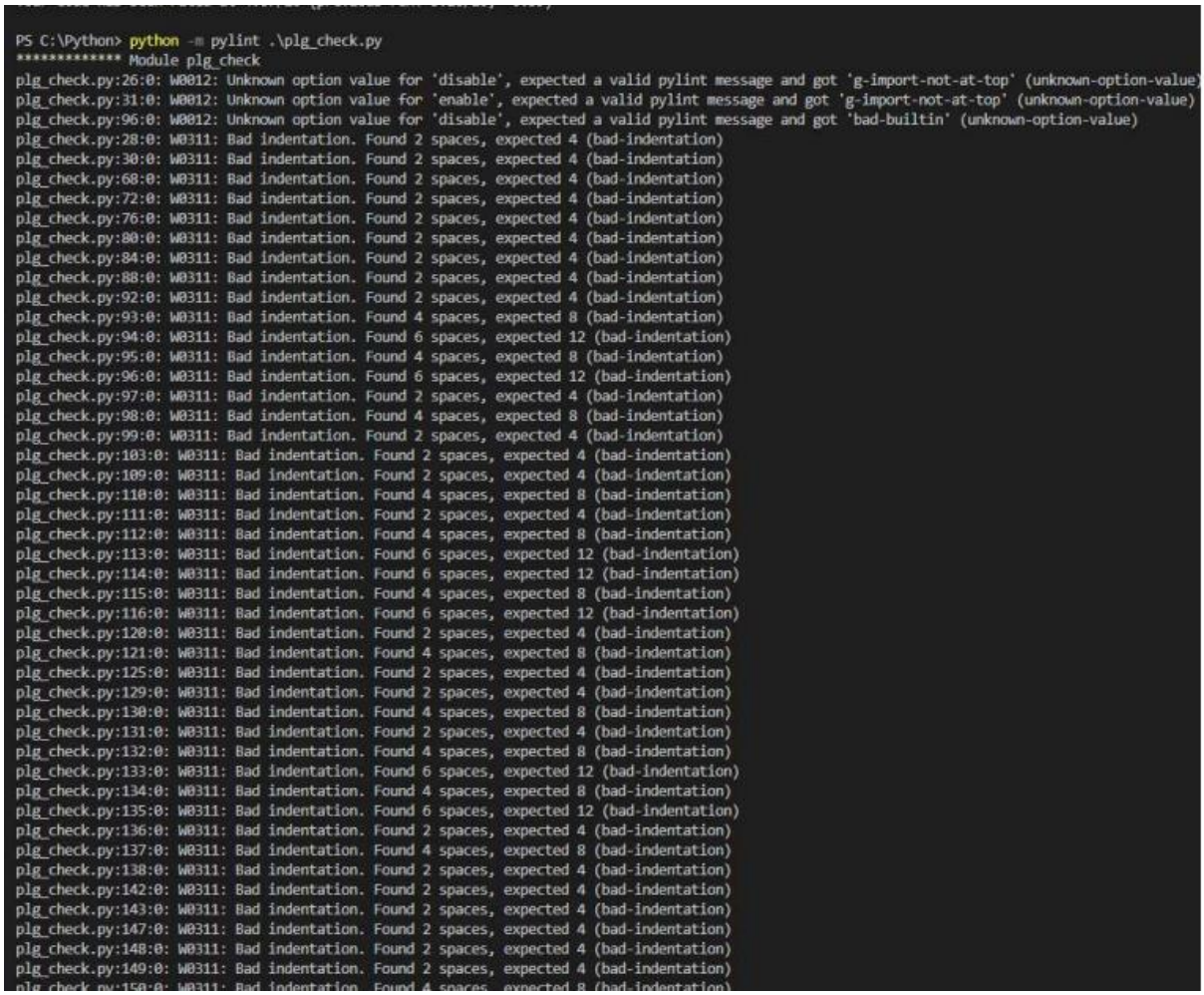


```
PS C:\Python> python -m pylint .\plg_check.py
************* Module plg_check
plg_check.py:49:0: C0301: Line too long (106/100) (line-too-long)
plg_check.py:53:0: C0301: Line too long (113/100) (line-too-long)
plg_check.py:54:0: C0301: Line too long (113/100) (line-too-long)
plg_check.py:62:0: C0301: Line too long (105/100) (line-too-long)
plg_check.py:63:0: C0301: Line too long (113/100) (line-too-long)
plg_check.py:64:0: C0301: Line too long (108/100) (line-too-long)
plg_check.py:68:0: C0301: Line too long (110/100) (line-too-long)
plg_check.py:117:0: C0301: Line too long (111/100) (line-too-long)
plg_check.py:125:0: C0301: Line too long (112/100) (line-too-long)
plg_check.py:126:0: C0301: Line too long (113/100) (line-too-long)
plg_check.py:173:0: C0304: Final newline missing (missing-final-newline)
plg_check.py:1:0: C0114: Missing module docstring (missing-module-docstring)
plg_check.py:11:0: E0401: Unable to import 'openai' (import-error)
plg_check.py:12:0: E0401: Unable to import 'tqdm' (import-error)
plg_check.py:13:0: E0401: Unable to import 'openai' (import-error)
plg_check.py:16:0: C0103: Type alias name "StrOrOpenAIObject" doesn't conform to predefined naming style (invalid-name)
plg_check.py:21:4: W1203: Use lazy % formatting in logging functions (logging-fstring-interpolation)
plg_check.py:25:0: C0115: Missing class docstring (missing-class-docstring)
plg_check.py:29:4: C0103: Attribute name "n" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:25:0: R0205: Class 'OpenAIDecodingArguments' inherits from object, can be safely removed from bases in python3
plg_check.py:25:0: R0902: Too many instance attributes (11/7) (too-many-instance-attributes)
plg_check.py:39:0: R0913: Too many arguments (8/5) (too-many-arguments)
plg_check.py:39:0: R0914: Too many local variables (21/15) (too-many-locals)
plg_check.py:101:32: R1735: Consider using '{"model": model_name, **batch_decoding_args.__dict__, **decoding_kwargs, ... }'
plg_check.py:113:12: C0103: Variable name "e" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:114:16: W1203: Use lazy % formatting in logging functions (logging-fstring-interpolation)
plg_check.py:117:20: W1203: Use lazy % formatting in logging functions (logging-fstring-interpolation)
plg_check.py:133:20: C0103: Argument name "f" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:138:12: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
plg_check.py:138:12: R1732: Consider using 'with' for resource-allocating operations (consider-using-with)
plg_check.py:142:20: C0103: Argument name "f" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:144:12: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
plg_check.py:144:12: R1732: Consider using 'with' for resource-allocating operations (consider-using-with)
plg_check.py:148:15: C0103: Argument name "f" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:168:10: C0103: Argument name "f" doesn't conform to snake_case naming style (invalid-name)
plg_check.py:14:0: C0411: standard import "import copy" should be placed before "import openai" (wrong-import-order)
```

Code:

```
    if ',' in syslibs:
      syslibs = ','.join(sorted(syslibs.split(',')))
    else:
      syslibs = ','.join(sorted(syslibs.split()))
    write_action_env_to_bazelrc('TF_SYSTEM_LIBS', syslibs)

  for varname in ('PREFIX', 'LIBDIR', 'INCLUDEDIR', 'PROTOBUF_INCLUDE_PATH'):
    if varname in environ_cp:
      write_to_bazelrc('build --define=%s=%s' % (varname, environ_cp[varname]))


def set_windows_build_flags(environ_cp):
  """Set Windows specific build options."""

  # First available in VS 16.4. Speeds up Windows compile times by a lot. See
  # https://groups.google.com/a/tensorflow.org/d/topic/build/SsW98Eo7l3o/discussion
  # pylint: disable=line-too-long
  write_to_bazelrc(
      'build --copt=/d2ReducedOptimizeHugeFunctions --host_copt=/d2ReducedOptimizeHugeFunctions'
  )

  if get_var(
      environ_cp, 'TF_OVERRIDE_EIGEN_STRONG_INLINE', 'Eigen strong inline',
      True, ('Would you like to override eigen strong inline for some C++ '
             'compilation to reduce the compilation time?'),
      'Eigen strong inline overridden.', 'Not overriding eigen strong inline, '
      'some compilations could take more than 20 mins.'):
    # Due to a known MSVC compiler issue
    # https://github.com/tensorflow/tensorflow/issues/10521
    # Overriding eigen strong inline speeds up the compiling of
    # conv_grad_ops_3d.cc and conv_ops_3d.cc by 20 minutes,
    # but this also hurts the performance. Let users decide what they want.
    write_to_bazelrc('build --define=override_eigen_strong_inline=true')


def config_info_line(name, help_text):
  """Helper function to print formatted help text for Bazel config options."""
  print('\t--config=%-12s\t# %s' % (name, help_text))


def configure_ios(environ_cp):
  """Configures TensorFlow for iOS builds."""
  if not is_macos():
    return
  if not get_var(environ_cp, 'TF_CONFIGURE_IOS', 'iOS', False):
```

Static Analysis:



```
PS C:\Python> python -m pylint .\plg_check.py
************* Module plg_check
plg_check.py:26:0: W0012: Unknown option value for 'disable', expected a valid pylint message and got 'g-import-not-at-top' (unknown-option-value)
plg_check.py:31:0: W0012: Unknown option value for 'enable', expected a valid pylint message and got 'g-import-not-at-top' (unknown-option-value)
plg_check.py:96:0: W0012: Unknown option value for 'disable', expected a valid pylint message and got 'bad-builtin' (unknown-option-value)
plg_check.py:28:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:30:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:68:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:72:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:76:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:80:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:84:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:88:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:92:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:93:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:94:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:95:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:96:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:97:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:98:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:99:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:103:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:109:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:110:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:111:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:112:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:113:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:114:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:115:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:116:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:120:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:121:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:125:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:129:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:130:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:131:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:132:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:133:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:134:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:135:0: W0311: Bad indentation. Found 6 spaces, expected 12 (bad-indentation)
plg_check.py:136:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:137:0: W0311: Bad indentation. Found 4 spaces, expected 8 (bad-indentation)
plg_check.py:138:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:142:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:143:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:147:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:148:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
plg_check.py:149:0: W0311: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
```

We have used pylint Python tool to do Static analysis.

- At the end of analysis rate has been given to each code.
- Missing class docstring, import error, missing module, wrong import order, invalid name, global variable not assigned, invalid name, bad-identation this are the errors detected by pylint tool.