Copyleaks

Scan details

Scan Time: April 5th, 2025 at 05:12 UTC

Total Pages: 2

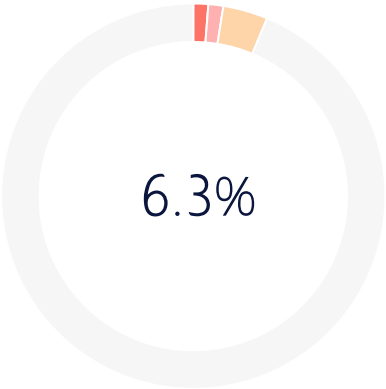
Total Words: 478

Analysis Report

Plagiarism Detection and AI Detection Report

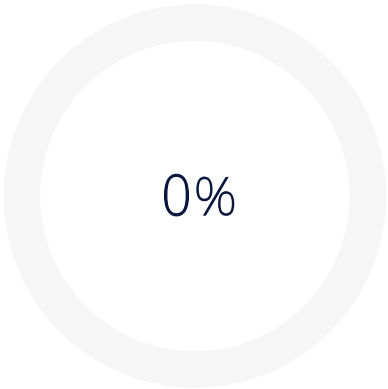
New Scan 10:42

Plagiarism Detection



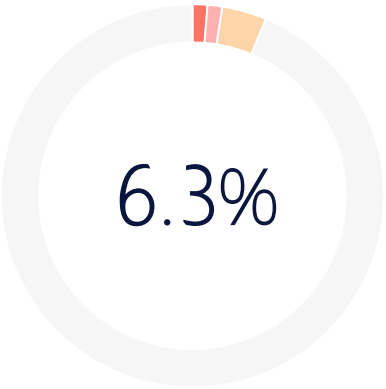
Plagiarism Types	Text Coverage	Words
Identical	1.3%	6
Minor Changes	1.3%	6
Paraphrased	3.8%	18
Excluded		
Omitted Words		0

AI Detection



	Text Coverage	Words
AI Text	0%	0
Human Text	100%	478
Excluded		
Omitted Words		0

Plagiarism



Results (5)

*Results may not appear because the feature has been disabled.

Repository	Internal Database	Filtered / Excluded
0	0	0

Internet Sources	Current Batch
5	0

Plagiarism Types	Text Coverage	Words
Identical	1.3%	6
Minor Changes	1.3%	6
Paraphrased	3.8%	18
Excluded		
Omitted Words		0

About Plagiarism Detection

Our AI-powered plagiarism scans offer three layers of text similarity detection: Identical, Minor Changes, and Paraphrased. Based on your scan settings we also provide insight on how much of the text you are not scanning for plagiarism (Omitted words).

Identical

One to one exact word matches. Learn more

Minor Changes

Words that hold nearly the same meaning but have a change to their form (e.g. "large" becomes "largely"). Learn more

Paraphrased

Different words that hold the same meaning that replace the original content (e.g. 'large' becomes 'big') Learn more

Omitted Words

The portion of text that is not being scanned for plagiarism based on the scan settings. (e.g. the 'Ignore quotations' setting is enabled and the document is 20% quotations making the omitted words percentage 20%) Learn more

Copyleaks Internal Database

Our Internal Database is a collection of millions of user-submitted documents that you can utilize as a scan resource and choose whether or not you would like to submit the file you are scanning into the Internal Database. Learn more

Filtered and Excluded Results

The report will generate a complete list of results. There is always the option to exclude specific results that are not relevant. Note, by unchecking certain results, the similarity percentage may change. Learn more

Current Batch Results

These are the results displayed from the collection, or batch, of files uploaded for a scan at the same time. Learn more

Plagiarism Detection Results: (5)

Creating a Camera Application using Pyqt5 — 6.3%

https://www.tutorialspoint.com/creating-a-camera-application-using-pyqt5

Home Online Compilers Wh...

---

main.py — 2.5%

https://raw.githubusercontent.com/songjiahao-wq/pyqt5-yolov5-7.0-2/master/main.py

from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QMenu, QAction from main_win.win import Ui_mainWindow from PyQt5....

---

main.py — 2.5%

https://raw.githubusercontent.com/javacr/pyqt5-yolov5/yolov5_v6.1/main.py

from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QMenu, QAction from main_win.win import Ui_mainWindow from PyQt5....

---

python - PyQt QThread Segfault while streaming a video - Stack Overflow — 2.5%

https://stackoverflow.com/questions/73971696/pyqt-qthread-segfault-while-streaming-a-video

Skip to main content Stack Overflow ...

---

import sys import cv2 import numpy as np from PyQt5.QtCore import QTimer, Qt ... — 2.5%

https://wenku.csdn.net/answer/841d6sx1y4

首页import sys import cv2 import numpy as np from PyQt5.QtCore import QTimer, Qt from PyQt5.QtGui import QImage, QPixmap from PyQt5.Qt...

```
import sys
import torch
import cv2
import ollama
import sounddevice as sd
import numpy as np
import speech_recognition as sr
from PyQt5.QtCore import QThread, pyqtSignal
from PyQt5.QtGui import QImage, QPixmap
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QPushButton, QTextEdit, QLabel
from yolov5 import YOLOv5

class AudioRecorderThread(QThread):
text_update = pyqtSignal(str)
finished_recording = pyqtSignal(str)

def __init__(self):
super().__init__()
self.sample_rate = 16000
self.recognizer = sr.Recognizer()
self.audio_buffer = []
self.running = False

def run(self):
self.running = True
self.audio_buffer.clear()

print('Listening...')
stream = sd.InputStream(
samplerate=self.sample_rate,
channels=1,
dtype=np.int16,
callback=self.audio_callback
)

with stream:
while self.running:
```

```
sd.sleep(500)

if self.audio_buffer:
self.process_transcription()a

def audio_callback(self, indata, frames, time, status):
if self.running:
self.audio_buffer.append(indata.copy())

def process_transcription(self):
'''Convert recorded audio into text and send to chatbot.'''
audio_data = np.concatenate(self.audio_buffer, axis=0)
audio_data = np.array(audio_data, dtype=np.int16)
audio = sr.AudioData(audio_data.tobytes(), self.sample_rate, 2)

try:
print('Transcribing...')
text = self.recognizer.recognize_google(audio)
self.text_update.emit(text)
self.finished_recording.emit(text) # Send text to chatbot
except sr.UnknownValueError:
self.text_update.emit('Could not understand the speech.')
self.finished_recording.emit('')
except sr.RequestError:
self.text_update.emit('Speech Recognition service error.')
self.finished_recording.emit('')

def stop(self):
self.running = False
self.wait()

class VideoCaptureThread(QThread):
frame_captured = pyqtSignal(QImage)

def __init__(self):
super().__init__()
self.capture = cv2.VideoCapture(0)
self.yolo_model = YOLOv5('yolov5s.pt', device=torch.device('cuda' if torch.cuda.is_available() else 'cpu'))
self.running = True
```

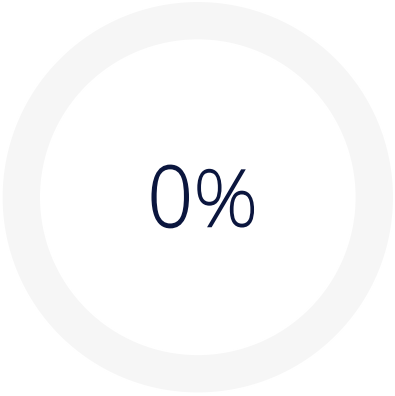
```
def run(self):
while self.running:
ret, frame = self.capture.read()
if ret:
results = self.yolo_model.predict(frame)
frame_with_boxes = results.render()[0]

height, width, _ = frame_with_boxes.shape
q_image = QImage(frame_with_boxes.data, width, height, 3 * width, QImage.Format_BGR888)
self.frame_captured.emit(q_image)

def stop(self):
self.running = False
self.capture.release()
self.wait()

class SmartClassroomAssistant(QWidget):
def __init__(self):
super().__init__()
self.initUI()
self.audio_thread = None
self.video_thread = VideoCaptureThread()
self.video_thread.frame_captured.connect(self.update_frame)
self.video_thread.start()
self.chat_history = []

def initUI(self):
self.setWindowTitle('Smart Classroom Assistant - AI Chatbot & Real-Time Detection')
self.setGeometry(100, 100, 800, 600)

layout = QVBoxLayout()

self.video_label = QLabel(self)
layout.addWidget(self.video_label)

self.transcription_display = QTextEdit(self)
self.transcription_display.setReadOnly(True)
self.transcription_display.setPlaceholderText('Transcription will appear here...')
layout.addWidget(self.transcription_display)
```

```
self.chat_display = QTextEdit(self)
self.chat_display.setReadOnly(True)
self.chat_display.setPlaceholderText('Assistant responses will appear here...')
layout.addWidget(self.chat_display)

self.push_to_talk_btn = QPushButton('🎤 Start Listening', self)
self.push_to_talk_btn.clicked.connect(self.start_recording)
layout.addWidget(self.push_to_talk_btn)

self.stop_listening_btn = QPushButton('🛑 Stop Listening', self)
self.stop_listening_btn.clicked.connect(self.stop_recording)
self.stop_listening_btn.setEnabled(False)
layout.addWidget(self.stop_listening_btn)

self.stop_btn = QPushButton('Exit', self)
self.stop_btn.clicked.connect(self.stop_application)
layout.addWidget(self.stop_btn)

self.setLayout(layout)

def update_frame(self, q_image):
self.video_label.setPixmap(QPixmap.fromImage(q_image))

def update_transcription(self, text):
'''Update the transcription display.'''
self.transcription_display.setPlainText(f'Live Transcription:\n{text}')

def send_to_chatbot(self, text):
'''Send transcribed text to chatbot and display the response.'''
if text.strip():
self.chat_display.append(f'You: {text}')
bot_response = self.chat_with_ollama(text)
self.chat_display.append(f'Assistant: {bot_response}')

def start_recording(self):
'''Start real-time speech-to-text.'''
if self.audio_thread is not None:
self.audio_thread.stop()

self.audio_thread = AudioRecorderThread()
```

```
self.audio_thread.text_update.connect(self.update_transcription)
self.audio_thread.finished_recording.connect(self.send_to_chatbot) # Send transcript to chatbot

self.transcription_display.clear()
self.transcription_display.setPlaceholderText('Listening... Speak now')
self.push_to_talk_btn.setEnabled(False)
self.stop_listening_btn.setEnabled(True)
self.audio_thread.start()

def stop_recording(self):
'''Stop recording and process the final transcription.'''
if self.audio_thread:
self.audio_thread.stop()
self.push_to_talk_btn.setEnabled(True)
self.stop_listening_btn.setEnabled(False)
self.transcription_display.setPlaceholderText('Transcription will appear here...')

def chat_with_ollama(self, user_input, model='mistral'):
'''Send user input to the Ollama chatbot and get a response.'''
self.chat_history.append({'role': 'user', 'content': user_input})

try:
response = ollama.chat(model=model, messages=self.chat_history)
bot_reply = response.get('message', {}).get('content', '[No response received]')
self.chat_history.append({'role': 'assistant', 'content': bot_reply})
return bot_reply
except ollama._types.ResponseError as e:
return f'Ollama Error: {e}'
except Exception as e:
return f'Unexpected Error: {e}'

def stop_application(self):
'''Clean up and exit.'''
if self.audio_thread:
self.audio_thread.stop()
self.video_thread.stop()
self.close()

def closeEvent(self, event):
'''Cleanup when the window is closed.'''
```

```
if self.audio_thread:
self.audio_thread.stop()
self.video_thread.stop()
event.accept()

if __name__ == '__main__':
app = QApplication(sys.argv)
window = SmartClassroomAssistant()
window.show()
sys.exit(app.exec_())
```

AI Content



	Text Coverage	Words
AI Text	0%	0
Human Text	100%	478
Excluded		
Omitted Words		0

About AI Detection

Our AI Detector is the only enterprise-level solution that can verify if the content was written by a human or generated by AI, including source code and text that has been plagiarized or modified. Learn more

AI Text

A body of text that has been generated or altered by AI technology. Learn more

Human Text

Any text that has been fully written by a human and has not been altered or generated by AI. Learn more

Copyleaks AI Detector Effectiveness

Credible data at scale, coupled with machine learning and widespread adoption, allows us to continually refine and improve our ability to understand complex text patterns, resulting in over 99% accuracy–far higher than any other AI detector–and improving daily. Learn more

Ideal Text Length

The higher the character count, the easier for our technology to determine irregular patterns, which results in a higher confidence rating for AI detection. Learn more

Reasons It Might Be AI When You Think It's Not

The AI Detector can detect a variety of AI-generated text, including tools that use AI technology to paraphrase content, auto-complete sentences, and more. Learn more

User AI Alert History

Historical data of how many times a user has been flagged for potentially having AI text within their content. Learn more

AI Insights

The number of times a phrase was found more frequently in AI vs human text is shown according to low, medium, and high frequency. Learn more