

タブ 1

ROS noeticでOllama(GPU無し)を利用

目標は、デイサービスで働くロボット。今回製作するロボットは、利用者のところまで自走し、お喋りを始めることを目標とする。私にとっては、ハードルが高いかな？やれるだけやっていく。そして今回は、ROS noeticでOllamaを利用出来るようにすること。

構成

・W8yellow\$	Client PC CF-W8	Debian12
・Ryzen9\$	Server PC Minis Forum HX90	Debian12

目次

[概要](#)

[誤記の確認](#)

[docker run オプション: --net=host と -p](#)

[生成AIの入れ替え](#)

[質問内容の整理](#)

[ggml参考プログラム](#)

[プログラムの修正](#)

[依存ライブラリ](#)

[ros_ollamaパッケージの作成](#)

[nodeの作製](#)

[Launchファイルの作製](#)

[ollamaのdocker runコマンド](#)

[実行スクリプト](#)

[実行](#)

[結論](#)

[今後の予定](#)

[参考](#)

[回答: Gemini](#)

[回答: ChatGPT](#)

概要

やはり、過去の記事を読んでも、しっくりこない。

ほぼ、忘れている。数ヶ月前の話であるが。。とにかく、Ollamaを起動して、gemma3:12bを動かしてみる。重ければ、下げていく。27bについては、GPU無しなので、テスト対象外とする。いや、最後にやるだけやってみるか？

誤記の確認

前回の記事：[20250529人型RB noeticでollama API回答生成](#) を実行中に気がついた誤記について記載する。

docker run オプション: --net=host と -p

誤記の場所：[20250529人型RB noeticでollama API回答生成](#)

ros runのオプションで、--net=hostと-p が併用されている。現在の私の理解では、--net=hostオプションを指定した段階で、ネットの環境はhostと共有である。なので、-pオプションが意味をなさない。ホストPCでコンテナのポート番号を利用しないのであれば、--net=hostのほうが、パフォーマンスは良い。あとはセキュリティだが、今回のシステムは、完成後はインターネットに接続しない。開発環境では-pを使い、実行環境では--net=hostで良いと考える。

他にもエラーが潜んでいるかもしれない。発見したらここに追記していく。

生成AIの入れ替え

改めて、生成AIにプログラムを作成してもらう。

質問内容の整理

Ros noeticを使ってロボットの製作を行っている。

- ・ホストPCはDebian12で、Ros noeticはDockerを使って稼働している。
- ・Ollamaは、DockerでOllama公式サイトから引用したimageで、同一ホストで稼働している。
- ・Ollamaへのテキストは、APIを利用し、質問を送信し回答を得る。
- ・Ollamaのモデルは、Gemma3n:e2bとする。
- ・Ollamaのdocker runコマンドにはオプションに"" -p 11434:11434"" をつけてあります。

・現在ggmlを使った参考プログラムがあるので、それをOllamaのAPIを使うように変更する。
以上の点に配慮し、プログラムの修正をしてください。

ggml参考プログラム

"""

```
#!/usr/bin/env python3
#catkin_ws/src/ros_ggml/scripts/chatgpt02_node.py
import rospy
from std_msgs.msg import String
import subprocess
from std_msgs.msg import Bool #20250120考え中の顔
from std_msgs.msg import Int32 #20250309腕の動き
```

#トピック名の設定

INPUT_TOPIC = "speech_recognition/final_result"

OUTPUT_TOPIC = "gpt_neox/result"

input_text = None

def process_input(data):

"""

受け取ったトピックデータを処理し、外部コマンドを実行して結果をパブリッシュする。

"""

global input_text

if data.data != input_text:

input_text = data.data

rospy.loginfo(f"Received input: {input_text}")

#20250120考え中の顔

bool_msg = Bool()

bool_msg.data = True

pub.publish(bool_msg)

#20250309腕の動き

bool_msg_arm_start = Bool()

bool_msg_arm_start.data = True

start_pub.publish(bool_msg_arm_start)

int32_msg_work_type = Int32()

```

int32_msg_work_type.data = 2 #腕を動かす
workout_pub.publish(int32_msg_work_type)
#20250312音声キャプチャ停止
bool_msg = Bool()
bool_msg.data = False
audio_pub.publish(bool_msg)

# ./bin/gpt-neox コマンドを実行
command = [

"/home/yo/catkin_ws/src/ros_ggml/scripts/ggml/build/bin/gpt-neox",
    "-n", "40",
#    "-m",
"/home/yo/workspace/models/ggml/line-corp-japanese-large-lm-3.6b
-ggml-q8_0.bin",
    "-m",
"/home/yo/workspace/models/ggml/line-corp-japanese-large-lm-3.6b
-ggml-q4_0.bin",
    "-p", input_text
]

try:
    # コマンドを実行して結果を取得
    result = subprocess.check_output(command, text=True)

    # 不要な行を削除
    filtered_result = "\n".join(
        line for line in result.splitlines()
        if not (line.startswith("gpt_") or line.startswith("main:") or
line.startswith("INFO"))
    )
    #「。」以降の文章を削除する。
    if "。" in filtered_result:
        filtered_result = filtered_result[:filtered_result.rfind("。
")+ 1]

    # フィルタ後の結果をログとトピックに出力
    #rospy.loginfo(f"Filtered output:
{filtered_result.replace('<0x0A>', ")}")

```

```

        rospy.loginfo(f"Filtered output:
{filtered_result.replace('<0x0A>', '').replace('</s>', '')}")
        output_publisher.publish(filtered_result.replace('<0x0A>',
'').replace('</s>', ''))

```

```

    except subprocess.CalledProcessError as e:
        rospy.logerr(f"Error while executing the command: {e}")
    else:
        rospy.loginfo(f"Received the same message: {input_text}")

```

```

def main():

```

```

    """

```

```

    メイン関数: ROSノードの初期化、購読者とパブリッシャの設定。

```

```

    """

```

```

    rospy.init_node("gpt_neox_node", anonymous=True)

```

```

    # パブリッシャの設定

```

```

    global output_publisher

```

```

    output_publisher = rospy.Publisher(OUTPUT_TOPIC, String,
queue_size=10)

```

```

    #20250120考え中の顔

```

```

    global pub

```

```

    pub = rospy.Publisher('face_eye_on', Bool, queue_size=10)

```

```

#20250120

```

```

    #20250309腕の動き

```

```

    global start_pub

```

```

    start_pub = rospy.Publisher("rb_work_out_start", Bool,
queue_size=10)

```

```

    global workout_pub

```

```

    workout_pub = rospy.Publisher("rb_work_out", Int32,
queue_size=10)

```

```

    #20250312音声キャプチャ停止

```

```

    global audio_pub

```

```

    audio_pub = rospy.Publisher("audio_on", Bool, queue_size=10)

```

```

    # サブスクライバの設定

```

```

    rospy.Subscriber(INPUT_TOPIC, String, process_input)

```

```
rospy.loginfo("gpt_neox_node is running...")
rospy.spin()
```

```
if __name__ == "__main__":
    main()
"""
```

回答は[参考](#)に示す。

プログラムの修正

ChatGPTの[回答](#)を参考に作業を進めていく。

依存ライブラリ

```
noetic yo@Ryzen9:~$ sudo apt install python3-pip
pip3 install requests
```

ros_ollamaパッケージの作成

```
noetic yo@Ryzen9:~/catkin_ws/src$ catkin_create_pkg ros_ollama
rospy roscpp std_msgs
noetic yo@Ryzen9:~/catkin_ws/src$ cd ..
noetic yo@Ryzen9:~/catkin_ws$ catkin_make
noetic yo@Ryzen9:~$ source ~/catkin_ws/devel/setup.bash
noetic yo@Ryzen9:~$ roscd ros_ollama
```

nodeの作製

```
#!/usr/bin/env python3
# catkin_ws/src/ros_ollama/scripts/chatgpt_ollama_node.py
```

```
import rospy
import requests
from std_msgs.msg import String, Bool, Int32
```

#トピック名の設定

```
INPUT_TOPIC = "speech_recognition/final_result"
OUTPUT_TOPIC = "gpt_neox/result"
input_text = None
```

```
OLLAMA_API_URL = "http://localhost:11434/api/generate"
```

```
OLLAMA_MODEL_NAME = "gemma3n:e2b"
```

```
def call_ollama_api(prompt):
```

```
    """
```

```
    OllamaのAPIにリクエストを送信して応答を取得
```

```
    """
```

```
    try:
```

```
        response = requests.post(OLLAMA_API_URL, json={
            "model": OLLAMA_MODEL_NAME,
            "prompt": prompt,
            "stream": False # ストリーミングでない一括応答
        })
```

```
        response.raise_for_status()
```

```
        data = response.json()
```

```
        return data.get("response", "")
```

```
    except requests.RequestException as e:
```

```
        rospy.logerr(f'Ollama API request failed: {e}')
        return "(Ollama APIエラー)"
```

```
def process_input(data):
```

```
    global input_text
```

```
    if data.data != input_text:
```

```
        input_text = data.data
```

```
        rospy.loginfo(f'Received input: {input_text}')
```

```
    # 考え中の顔を表示
```

```
    pub.publish(Bool(data=True))
```

```
    # 腕の動き開始
```

```
    start_pub.publish(Bool(data=True))
```

```
    workout_pub.publish(Int32(data=2)) # 腕を動かす動作
```

```
    # 音声キャプチャ停止
```

```
    audio_pub.publish(Bool(data=False))
```

```
    # Ollama APIで応答生成
```

```
    result = call_ollama_api(input_text)
```

```
    # フィルタ処理
```

```

        filtered_result = "\n".join(
            line for line in result.splitlines()
            if not (line.startswith("gpt_") or line.startswith("main:") or
line.startswith("INFO"))
        )

        # 「。」以降を削除
        if "。" in filtered_result:
            filtered_result = filtered_result[:filtered_result.rfind("。") + 1]

        filtered_result = filtered_result.replace('<0x0A>',
        ").replace('</s>', ")
        rospy.loginfo(f"Ollama output: {filtered_result}")
        output_publisher.publish(filtered_result)

    else:
        rospy.loginfo(f"Received the same message: {input_text}")

def main():
    rospy.init_node("gpt_ollama_node", anonymous=True)

    global output_publisher, pub, start_pub, workout_pub, audio_pub
    output_publisher = rospy.Publisher(OUTPUT_TOPIC, String,
queue_size=10)
    pub = rospy.Publisher("face_eye_on", Bool, queue_size=10)
    start_pub = rospy.Publisher("rb_work_out_start", Bool,
queue_size=10)
    workout_pub = rospy.Publisher("rb_work_out", Int32,
queue_size=10)
    audio_pub = rospy.Publisher("audio_on", Bool, queue_size=10)

    rospy.Subscriber(INPUT_TOPIC, String, process_input)
    rospy.loginfo("gpt_ollama_node is running...")
    rospy.spin()

if __name__ == "__main__":
    main()

```

Launchファイルの作製

```
<!-- ~/catkin_ws/src/ros_ollama/launch/all-rev1.launch -->
<launch>
  <!-- ros_voskノード 音声テキスト変換-->
  <node pkg="ros_vosk" type="chatgpt_node.py"
name="vosk_chatgpt_node" output="screen">
    <param name="vosk/model" value="vosk-model-small-ja-0.22" />
    <param name="audio_topic" value="/audio"/>
    <param name="model_path"
value="/home/yo/catkin_ws/src/ros-vosk/models"/>
    <param name="sample_rate" value="16000"/>
    <param name="max_alternatives" value="1"/>
    <param name="words" value="true"/>
    <param name="partial_results" value="false"/>
  </node>
  <!-- ~/catkin_ws/src/ros_vosk/scripts/chatgpt_node.py -->

  <!-- ollama_gemma27bノード 応答テキスト生成-->
  <node pkg="ros_ollama" type="ollama_gemma27b.py"
name="ollama_node" output="screen"> </node>
  <!-- ~/catkin_ws/src/ros_ollama/scripts/ollama_gemma27b.py -->

  <!-- pyjtalkノード テキスト音声変換-->
  <node pkg="ros_pyopenjtalk" type="chatgpt_pyopenjtalk01.py"
name="pyopenjtalk_node" output="screen"> </node>
  <!--
~/catkin_ws/src/ros_pyopenjtalk/scripts/chatgpt_pyopenjtalk01.py
-->
</launch>
```

ollamaのdocker runコマンド

```
yo@Ryzen9:~/command/12docker_ollama$ cat
docker_run_ollama.sh
#!/bin/bash
docker run -d \
  -v ollama:/root/.ollama \
  -p 11434:11434 \
```

```
--name ollama \  
ollama/ollama
```

前回利用したコマンド

```
docker run -d \  
--gpus all \           #GPUを使う  
--restart unless-stopped \  #自動再起動  
-v ollama_data:/root/.ollama \  
---net=host \          #不要 実行環境では、こちらを選択予  
定。  
-p 11434:11434 \  
--name ollama \  
ollama/ollama
```

実行スクリプト

```
noetic yo@Ryzen9:~$ cat start_all-ollama.sh  
#!/bin/bash  
source /opt/ros/noetic/setup.bash  
source ~/catkin_ws/devel/setup.bash  
#ros通信のための設定  
export ROS_MASTER_URI=http://192.168.11.8:11311  
export ROS_IP=192.168.11.8  
  
roslaunch ros_ollama all-rev1.launch  
# <!-- ~/catkin_ws/src/ros_ollama/launch/all-rev1.launch -->
```

実行

```
noetic yo@Ryzen9:~$ ./start_all-ollama.sh  
... logging to  
/home/yo/.ros/log/72bb3f8e-5e3b-11f0-9828-1c83413171d6/roslau  
ch-Ryzen9-3553.log  
Checking log directory for disk usage. This may take a while.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
2025-07-11 Friday 22:09:42
```

結論

gemma3:12bは、GPU無しで実行することができなかった。

上記を除いて、ダウンロードしたモデルのなかでサイズが最も大きいのは、gemma3n:e4bである。このモデルは実行可能である。よって、今回はe4bを使っていく。ここで、ollama runからモデルを使えるようになるまで、概ね20秒ほどかかる。起動時の自動化のとき、配慮する必要がある。

今後の予定

- ・生成AIをggmlからOllama に変更し、モデルをGemma3n:e4bとする。
<ここまでOK>

- ・テキスト音声出力をpyopenjtalkからVOICEVOXに変更する。

- ・再度、Ros noeticで音声テキスト変換: Wisperを利用できるか検討する。

- ・音声テキスト変換VOSKの辞書ファイルをsmallからnormalに変更する。

- ・音声テキスト変換されたテキストを評価するnodeを作成する。

「えっと」の削除、評価の後、AIに送るか「指示」に対して必要なnodeを起動するか決める。

- ・ロボットの顔を表示するシステムをqt5に変更する。

- ・利用者の顔を認識するシステムを構築する。

- ・移動装置を制作する。

- ・顔に向かって進む。

- ・認識した顔の方向に進むようにする。

- ・一定の場所に落ち着くようにする。

- ・認識した顔に、話しかけるようにする。

参考

 20250529人型RB noeticでollama API回答生成

回答: Gemini

ご提示いただいたggmlを使ったROSノードを、Ollama APIを利用するように修正します。

プログラム修正のポイント

1. **``subprocess.check\_output``の削除**: ggmlのバイナリを実行する部分を削除し、Ollama APIを呼び出すコードに置き換えます。
2. **``requests``ライブラリの利用**: Ollama APIとの通信にはPythonの``requests``ライブラリを使用します。
3. **Ollama APIエンドポイントの設定**: Dockerで公開されているOllamaのAPIエンドポイント(``http://localhost:11434/api/generate``)を設定します。
4. **モデル名の指定**: モデルとして``gemma:2b``を指定します。
5. **APIリクエストとレスポンスの処理**: Ollama APIへのリクエストボディを構築し、レスポンスから生成されたテキストを抽出します。
6. **エラーハンドリング**: API呼び出し時のエラーを適切に処理します。

修正後のプログラム

```
``python
#!/usr/bin/env python3
#catkin_ws/src/ros_ggml/scripts/chatgpt02_node.py
import rospy
from std_msgs.msg import String
import requests # requestsライブラリを追加
from std_msgs.msg import Bool
from std_msgs.msg import Int32

#トピック名の設定
INPUT_TOPIC = "speech_recognition/final_result"
OUTPUT_TOPIC = "gpt_neox/result" #トピック名はそのまま維持
input_text = None

# Ollamaの設定
OLLAMA_API_URL = "http://localhost:11434/api/generate"
```

OLLAMA_MODEL = "gemma:2b" # 使用するモデルをGemma3n:e2b (gemma:2b) に設定

```
def process_input(data):
```

```
    """
```

受け取ったトピックデータを処理し、Ollama APIに質問を送信して結果をパブリッシュする。

```
    """
```

```
    global input_text
```

```
    if data.data != input_text:
```

```
        input_text = data.data
```

```
        rospy.loginfo(f"Received input: {input_text}")
```

```
    # 考え中の顔
```

```
    bool_msg = Bool()
```

```
    bool_msg.data = True
```

```
    pub.publish(bool_msg)
```

```
    # 腕の動き
```

```
    bool_msg_arm_start = Bool()
```

```
    bool_msg_arm_start.data = True
```

```
    start_pub.publish(bool_msg_arm_start)
```

```
    int32_msg_work_type = Int32()
```

```
    int32_msg_work_type.data = 2 # 腕を動かす
```

```
    workout_pub.publish(int32_msg_work_type)
```

```
    # 音声キャプチャ停止
```

```
    bool_msg = Bool()
```

```
    bool_msg.data = False
```

```
    audio_pub.publish(bool_msg)
```

```
    try:
```

```
        # Ollama APIへのリクエストボディを構築
```

```
        payload = {
```

```
            "model": OLLAMA_MODEL,
```

```
            "prompt": input_text,
```

```
            "stream": False # ストリーミングではなく、完全な応答を受け
```

取る

```
        }
```

```

# Ollama APIにPOSTリクエストを送信
response = requests.post(OLLAMA_API_URL,
json=payload)
response.raise_for_status() # HTTPエラーがあれば例外を発生させる

```

```

# JSONレスポンスから生成されたテキストを抽出
result_json = response.json()
generated_text = result_json.get("response", "").strip()

```

「。」以降の文章を削除する。(Ollamaの応答に合わせて調整が必要な場合もあります)

```

if "." in generated_text:
    filtered_result = generated_text[:generated_text.rfind(".")]
+ 1]
else:
    filtered_result = generated_text # 「。」がない場合はそのまま

```

```

# フィルタ後の結果をログとトピックに出力
rospy.loginfo(f"Filtered output from Ollama: {filtered_result}")
output_publisher.publish(filtered_result)

```

```

except requests.exceptions.RequestException as e:
    rospy.logerr(f"Error while communicating with Ollama API: {e}")
except Exception as e:
    rospy.logerr(f"An unexpected error occurred: {e}")
else:
    rospy.loginfo(f"Received the same message: {input_text}")

```

```

def main():
    """
    メイン関数: ROSノードの初期化、購読者とパブリッシャの設定。
    """
    rospy.init_node("gpt_neox_node", anonymous=True)

    # パブリッシャの設定

```

```

global output_publisher
output_publisher = rospy.Publisher(OUTPUT_TOPIC, String,
queue_size=10)
# 考え中の顔
global pub
pub = rospy.Publisher('face_eye_on', Bool, queue_size=10)
# 腕の動き
global start_pub
start_pub = rospy.Publisher("rb_work_out_start", Bool,
queue_size=10)
global workout_pub
workout_pub = rospy.Publisher("rb_work_out", Int32,
queue_size=10)
# 音声キャプチャ停止
global audio_pub
audio_pub = rospy.Publisher("audio_on", Bool, queue_size=10)

# サブスクリバの設定
rospy.Subscriber(INPUT_TOPIC, String, process_input)

rospy.loginfo("gpt_neox_node (using Ollama) is running...")
rospy.spin()

if __name__ == "__main__":
    main()
...

```

修正後のプログラムを実行するための準備

1. ``requests`ライブラリのインストール``:

ROS環境のPythonに`requests`ライブラリがインストールされていない場合は、インストールする必要があります。Dockerコンテナ内でRos Noeticが稼働している場合、そのコンテナ内でインストールしてください。

```
``bash
```

```
pip install requests
```

```
...
```

または、Dockerイメージを再構築する際にDockerfileに`RUN pip install requests`を追加してください。

2. **Ollamaモデルのダウンロード**:

OllamaのDockerコンテナが起動していることを確認し、指定したモデル(`gemma:2b`)をダウンロードしておいてください。

```
```bash
```

```
docker exec -it <your_ollama_container_name_or_id> ollama pull
gemma:2b
```

```
...
```

`gemma3n:e2b`という表記はOllamaのモデル名としては一般的ではないため、おそらく`gemma:2b`を指しているものと解釈しました。もし別のモデル名であれば、`OLLAMA\_MODEL`変数を適切に変更してください。

## 3. **ROSパッケージの更新**:

修正したPythonファイルを

`catkin\_ws/src/ros\_ggml/scripts/chatgpt02\_node.py`に保存し、実行権限を付与してください。

```
```bash
```

```
chmod +x catkin_ws/src/ros_ggml/scripts/chatgpt02_node.py
```

```
...
```

これで、ROSノードがOllama APIを通じて応答を生成するようになります。

回答: ChatGPT

以下は、**ggmlのコマンドライン呼び出し部分を、OllamaのAPI呼び出しに置き換えたROSノード**です。

Ollamaは `http://localhost:11434/api/generate` に対してPOSTリクエストで問い合わせる仕様で、`model` に `"gemma3n:e2b"` を指定します。

修正ポイント

- * `subprocess` を使ったコマンド実行 → `requests` を使ってOllamaのREST APIへPOST

- * モデル名 `"gemma3n:e2b"` は明示的にAPIで指定

- * 出力整形(フィルター)は元のコードの処理を残してあります

依存ライブラリ

```
```bash
sudo apt install python3-pip
pip3 install requests
```
```

修正済みプログラム(`chatgpt\_ollama\_node.py`などとして保存)

```
```python
#!/usr/bin/env python3
catkin_ws/src/ros_ggml/scripts/chatgpt_ollama_node.py
```

```
import rospy
import requests
from std_msgs.msg import String, Bool, Int32
```

```
トピック名の設定
```

```
INPUT_TOPIC = "speech_recognition/final_result"
```

```
OUTPUT_TOPIC = "gpt_neox/result"
```

```
input_text = None
```

```
OLLAMA_API_URL = "http://localhost:11434/api/generate"
```

```
OLLAMA_MODEL_NAME = "gemma3n:e2b"
```

```
def call_ollama_api(prompt):
```

```
 """
```

```
 OllamaのAPIにリクエストを送信して応答を取得
```

```
 """
```

```
 try:
```

```
 response = requests.post(OLLAMA_API_URL, json={
 "model": OLLAMA_MODEL_NAME,
 "prompt": prompt,
 "stream": False # ストリーミングでない一括応答
 })
```

```
 response.raise_for_status()
```

```
 data = response.json()
```

```
 return data.get("response", "")
```

```
 except requests.RequestException as e:
```

```
 rospy.logerr(f'Ollama API request failed: {e}')
 return "(Ollama APIエラー)"
```

```
def process_input(data):
```

```
 global input_text
```

```
 if data.data != input_text:
```

```
 input_text = data.data
```

```
 rospy.loginfo(f'Received input: {input_text}')
```

```
 # 考え中の顔を表示
```

```
 pub.publish(Bool(data=True))
```

```
 # 腕の動き開始
```

```
 start_pub.publish(Bool(data=True))
```

```
 workout_pub.publish(Int32(data=2)) # 腕を動かす動作
```

```
 # 音声キャプチャ停止
```

```
 audio_pub.publish(Bool(data=False))
```

```
 # Ollama APIで応答生成
```

```
 result = call_ollama_api(input_text)
```

```

フィルタ処理
filtered_result = "\n".join(
 line for line in result.splitlines()
 if not (line.startswith("gpt_") or line.startswith("main:") or
line.startswith("INFO")))
)

「。」以降を削除
if "。" in filtered_result:
 filtered_result = filtered_result[:filtered_result.rfind("。") + 1]

 filtered_result = filtered_result.replace('<0x0A>',
 ").replace('</s>', ")
 rospy.loginfo(f"Ollama output: {filtered_result}")
 output_publisher.publish(filtered_result)

else:
 rospy.loginfo(f"Received the same message: {input_text}")

def main():
 rospy.init_node("gpt_ollama_node", anonymous=True)

 global output_publisher, pub, start_pub, workout_pub, audio_pub
 output_publisher = rospy.Publisher(OUTPUT_TOPIC, String,
queue_size=10)
 pub = rospy.Publisher("face_eye_on", Bool, queue_size=10)
 start_pub = rospy.Publisher("rb_work_out_start", Bool,
queue_size=10)
 workout_pub = rospy.Publisher("rb_work_out", Int32,
queue_size=10)
 audio_pub = rospy.Publisher("audio_on", Bool, queue_size=10)

 rospy.Subscriber(INPUT_TOPIC, String, process_input)
 rospy.loginfo("gpt_ollama_node is running...")
 rospy.spin()

if __name__ == "__main__":

```

```
main()
...
```

### ### テスト方法

1. `Ollama` コンテナが `http://localhost:11434` で稼働していることを確認
2. ROSノードを起動(例: `roslaunch ros\_ggml chatgpt\_ollama\_node.py`)
3. `rostopic pub /speech\_recognition/final\_result std\_msgs/String "こんにちは" `などで動作確認

必要に応じて、出力整形(例: 不要語除去・句点で切る処理)を強化できます。

また、Ollamaの応答が遅い場合、非同期化なども可能です。ご希望があれば対応案を提案します。