

Title

Generative AI for Humanitarian Cause: Schizophrenia Patient Data Analysis



Goal.....2

The primary goal is to develop a Schizophrenia Monitoring & Analysis System that leverages advanced AI technologies to assist medical professionals by providing a more standardized, objective, and efficient method of diagnosing and monitoring schizophrenic patients..... 2

This application will analyze multimodal data inputs - video recordings, speech patterns, and textual information from patient interactions to detect and predict symptom patterns and severity, aiding medical professionals in making informed treatment decisions..... 2

Problem Description.....2

Schizophrenia is a complex and chronic mental health disorder marked by symptoms such as delusions, hallucinations, disorganized thinking and impaired social functioning, often co-occurs with depression, anxiety disorders, and substance abuse..... 2

The symptoms not only significantly impact an individual's functioning of daily life, but also pose a risk of harm to others..... 2

Early intervention, comprehensive treatment approaches, and support from healthcare professionals, family, and community resources are essential in addressing the needs of individuals living with schizophrenia, but there is no existing effective solution..... 2

Solution Overview.....	3
What is Generative AI ?.....	3
Key Ideas of Generative AI.....	4
How It Works.....	4
Everyday Examples.....	5
Generative AI Application Development Environment.....	5
First a resource group and then a workspace need to be created inside Azure.....	6
Next the Azure ML Instance should be created for developing Notebook.....	7
Next the required Text, Audio and Video file data need to be stored in corresponding Datasets.....	7
Generative AI R&D performed in Azure Environment.....	8
Generative AI Analysis Techniques.....	8
Advanced AI Technologies are used for Text Analysis.....	8
Application of AI for Prescriptive Analytics:.....	8
AI-driven Knowledge Management:.....	8
Generative AI Notebook Setup.....	9
Generative AI Programming Examples.....	9
Here goes the Example for analyzing sample Audio data in Azure Notebook.....	9
Sentiment Analysis, keyword Extraction and Summarization using OpenAI.....	14
Audio Analysis using OpenAI Whisper Model.....	16
Translation using OpenAI Model.....	18
Audio Waveform Analysis using HuggingFace Model.....	19
Create a Chat Agent using Intelligent Prompts in Azure.....	21
Setup a Develop Environment in local Machine and Push Code to Azure.....	22
Provision a Virtual Machine and import Code from Github.....	22
Run Application.....	22
Run App.....	22
User Interaction.....	23
Analyze Textual Data.....	26
Analyze Audio Data.....	27
Analyze Video Data.....	28

Goal

The primary goal is to develop a Schizophrenia Monitoring & Analysis System that leverages advanced AI technologies to assist medical professionals by providing a more standardized, objective, and efficient method of diagnosing and monitoring schizophrenic patients.

This application will analyze multimodal data inputs - video recordings, speech patterns, and textual information from patient interactions to detect and predict symptom patterns and severity, aiding medical professionals in making informed treatment decisions.

Problem Description

Schizophrenia is a complex and chronic mental health disorder marked by symptoms such as delusions, hallucinations, disorganized thinking and impaired social functioning, often co-occurs with depression, anxiety disorders, and substance abuse.

The symptoms not only significantly impact an individual's functioning of daily life, but also pose a risk of harm to others. Early intervention, comprehensive treatment approaches, and support from healthcare professionals, family, and community resources are essential in addressing the needs of individuals living with schizophrenia, but there is no existing effective solution.

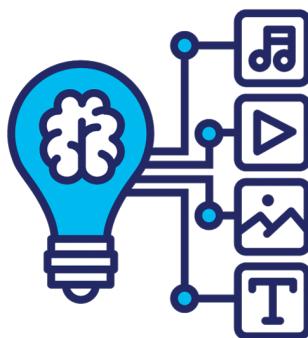
Solution Overview

Clinicians Use Cases



What is Generative AI ?

Generative AI is a class of artificial intelligence systems that can create new content. This can include text, images, music, and other types of data.



Key Ideas of Generative AI

1. **Creating New Stuff:** Think of generative AI as an incredibly smart and creative tool. It can write stories, draw pictures, compose music, and even design new products. It's like having an artist, writer, and inventor all rolled into one, powered by a computer.
2. **Learning from Examples:** Generative AI learns by studying lots of examples. For instance, if you wanted it to create new paintings, you'd show it thousands of pictures. It analyzes these to understand styles, colors, and shapes, and then uses that knowledge to make its own art.

How It Works

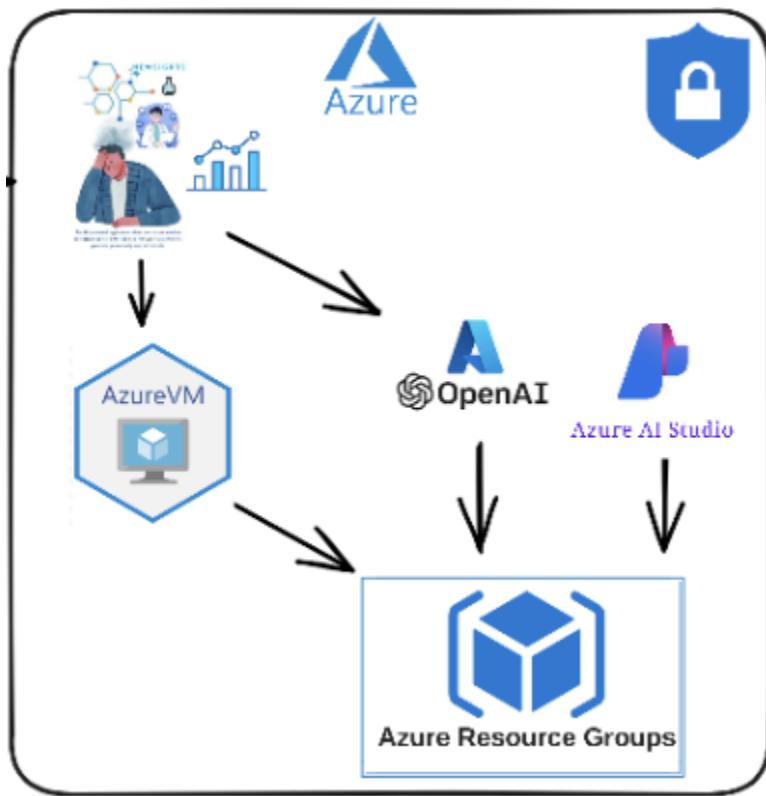
1. **Neural Networks:** These are computer systems inspired by the human brain. They process information in a way that's somewhat similar to how we do, helping the AI learn and create.
2. **Training:** The AI is trained on large datasets. If you want it to generate text, you feed it lots of books, articles, and other writings. It picks up on language patterns and styles from these examples.
3. **Generation:** Once trained, the AI can produce new content. For text, you might give it a starting sentence, and it will continue writing. For images, it might generate a completely new picture based on the styles it has learned.

Everyday Examples

- **Text Generation:** Tools like chatbots or writing assistants use generative AI to produce human-like text. They can help with drafting emails, writing articles, or even creating poetry.
- **Image Creation:** AI can create new artwork, design clothing, or even generate realistic photos of people who don't exist (like in deepfakes).
- **Music Composition:** Generative AI can compose original music, producing new songs that sound like they were created by a human musician.

Generative AI Application Development Environment

Azure Cloud was chosen as the Development Environment.



1. Azure Cognitive Services
 - Provides APIs for vision, speech, language, and decision-making capabilities.
 - [Learn more](#)
2. Azure Machine Learning
 - Platform for building, training, and deploying machine learning models.
 - Includes automated ML, designer interface, model management, and MLOps.
 - [Learn more](#)
3. Azure Bot Services
 - Tools for building, testing, and deploying conversational bots.
 - Integrates with multiple communication channels.
 - [Learn more](#)
4. Azure Cognitive Search
 - Cloud search service with AI capabilities for enriched indexing and semantic search.

- [Learn more](#)

5. Azure OpenAI Service

- Access to OpenAI's language models for natural language processing tasks.
- [Learn more](#)

6. Azure Data and AI Integration

- Seamless integration with Azure Synapse Analytics, Azure Data Factory, and Azure Databricks.
- [Learn more](#)

7. AI Infrastructure

- Specialized VMs, containers, AKS, and high-performance computing for AI workloads.
- [Learn more](#)

First a resource group and then a workspace need to be created inside Azure.

The screenshot shows the Azure portal interface for the 'srijon-workspace1' workspace. The top navigation bar includes 'Microsoft Azure', a search bar, and user information. The main content area shows the workspace name and type ('Azure Machine Learning workspace'). A sidebar on the left lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Monitoring, Automation, and Support + troubleshooting. The 'Overview' section is selected. The workspace details include:

- Resource group:** srijon-resourcegroup1
- Location:** West US
- Subscription:** Azure subscription 1
- Storage:** srijonworkspac0033153105
- Studio web URL:** <https://ml.azure.com?tid=c3d47919-b24a-4ba1-b4e4-86cf03...>
- Container Registry:** srijonworkspac4413913595
- Key Vault:** srijonworkspac1298975899
- Application Insights:** srijonworkspac1298975899
- MLflow tracking URI:** azureml://westus.api.azureml.ms/mlflow/v1.0/subscriptions/e...

Next the Azure ML Instance should be created for developing Notebook.

```

1 # Import packages used by the following code snippets
2 import csv
3 import json
4 import os
5 import requests
6 import time
7
8 import pandas as pd
9
10 from azure.ai.ml import Input, MLClient
11 from azure.ai.ml.constants import AssetTypes
12 from azure.identity import DefaultAzureCredential, InteractiveBrowserCredential
13 from azure.ai.ml.entities import (
14     AmlCompute,
15     BatchDeployment,
16     BatchEndpoint,
17     BatchRetrySettings,
18     Model,
19 )
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559

```

Next the required Text, Audio and Video file data need to be stored in corresponding Datasets.

Attribute	Value
Type	File (uri_file)
File URI	azureml://user_speech_file_v2:1
Created by	Srijon Mandal
Current version	1
Latest version	1
Created time	Apr 22, 2024 5:13 PM
Modified time	Apr 22, 2024 5:13 PM

Tags
No tags

Description
Click edit icon to add a description

Data sources
Datastore workspaceblobstore
Relative path UI/2024-04-23_001309_UTC/speech_sample.txt
Actions View in datastores browse View in Azure Portal
Datastore URI azureml://subscriptions/ed4b0003-99cf-4ec0-88ba-68847f627acf/resource... Storage URI https://srijonworkspac003153105.blob.core.windows.net/azureml-blobst...

Generative AI R&D performed in Azure Environment

Generative AI Analysis Techniques

Advanced AI Technologies are used for Text Analysis

- OpenAI Prompt Engineering and Whisper: For processing and understanding natural language inputs and converting speech to text with high accuracy.
- Hugging Face Transformer and OpenAI for Emotion Detection: To interpret emotional states from text, providing insights into the patient's mental state.
- Emotion Detection from Video using YOLOv7 and PyTorch: To analyze facial expressions and body language from video data, offering another layer of emotional and symptomatic analysis.

Application of AI for Prescriptive Analytics:

- ML Algorithms will be used to perform predictions and generate recommendations.

AI-driven Knowledge Management:

- Open-AI and RAG based Virtual Agent: It assists researchers and medical professionals by answering queries related to the patient data and broader schizophrenia research, facilitating an interactive, learning-enhanced environment.

Generative AI Notebook Setup

Once the Azure ML Studio instance is up and running in your workspace, its time to create the Notebooks.

```

try:
    credential = DefaultAzureCredential()
    credential.get_token("https://management.azure.com/.default")
except Exception as ex:
    credential = InteractiveBrowserCredential()

ml_client = MLClient(
    credentials=credential,
    subscription_id="ed4b0003-99cf-4ec0-88ba-68847f627acf",
    resource_group_name="srijon-resourcegroup1",
    workspace_name="srijon-workspace1",
)

# the models, fine tuning pipelines and environments are available in the AzureML system registry, "azureml"
registry_ml_client = MLClient(credential, registry_name="azurerm")

testdata = requests.get(
    "https://datasets-server.huggingface.co/first-rows?dataset=librispeech_asr&config=clean&split=test&off"
).text
testdata = json.loads(testdata)

model_name = "openai-whisper-large"
model_version = "10"
foundation_model = registry_ml_client.models.get(model_name, model_version)
print(foundation_model)

```

Generative AI Programming Examples

Here goes the Example for analyzing sample Audio data in Azure Notebook

- In this example,
- we import required dependencies, create an instance of `MLClient`,
- read the dataset into a `dataframe`,
- create a batch deployment endpoint
- perform inference
- store the prediction result
- terminate the inference cluster

```

# Import packages used by the following code snippets
import csv
import json
import os
import requests
import time

import pandas as pd

from azure.ai.ml import Input, MLClient
from azure.ai.ml.constants import AssetTypes

```

```
from azure.identity import DefaultAzureCredential,
InteractiveBrowserCredential
from azure.ai.ml.entities import (
    AmlCompute,
    BatchDeployment,
    BatchEndpoint,
    BatchRetrySettings,
    Model,
)
from azure.identity import (
    DefaultAzureCredential,
    InteractiveBrowserCredential,
    ClientSecretCredential,
)
from azure.ai.ml.entities import AmlCompute
import time

try:
    credential = DefaultAzureCredential()
    credential.get_token("https://management.azure.com/.default")
except Exception as ex:
    credential = InteractiveBrowserCredential()

ml_client = MLClient(
    credential=credential,
    subscription_id="ed4b0003-99cf-4ec0-88ba-68847f627acf",
    resource_group_name="srijon-resourcegroup1",
    workspace_name="srijon-workspace1",
)

# the models, fine tuning pipelines and environments are available in the
# AzureML system registry, "azureml"
registry_ml_client = MLClient(credential, registry_name="azureml")

model_name = "openai-whisper-large"
model_version = "10"
foundation_model = registry_ml_client.models.get(model_name,
model_version)
print()
```

```

f"Using model name: {foundation_model.name}, version:
{foundation_model.version}, id: {foundation_model.id} for inferencing."
)

audio_urls_and_text = [
    (row["row"]["audio"][0]["src"], row["row"]["text"]) for row in
testdata["rows"]
]
testdata = pd.DataFrame(data=audio_urls_and_text, columns=["audio",
"text"])
# Define directories and filenames as variables
dataset_dir = "librispeech-dataset"
test_datafile = "test_100.csv"

batch_dir = "batch"
batch_inputs_dir = os.path.join(batch_dir, "inputs")
batch_input_file = "batch_input.csv"
os.makedirs(dataset_dir, exist_ok=True)
os.makedirs(batch_dir, exist_ok=True)
os.makedirs(batch_inputs_dir, exist_ok=True)
pd.set_option(
    "display.max_colwidth", 0
) # Set the max column width to 0 to display the full text
test_df.head()

batch_df = test_df[["audio", "language"]]

# Divide this into files of 10 rows each
batch_size_per_predict = 10
for i in range(0, len(batch_df), batch_size_per_predict):
    j = i + batch_size_per_predict
    batch_df[i:j].to_csv(
        os.path.join(batch_inputs_dir, str(i) + batch_input_file),
        quoting=csv.QUOTE_ALL
    )

# Check out the first and last file name created
input_files = os.listdir(batch_inputs_dir)
print(f"{input_files[0]} to {str(i)}{batch_input_file}.")
```

compute_name = "test-cpu-cluster"

```

compute_cluster = AmlCompute(
    name=compute_name,
    description="An AML compute cluster",
    size="Standard_E4as_v4",
    min_instances=1,
    max_instances=2,
    idle_time_before_scale_down=120,
) # 120 seconds

ml_client.begin_create_or_update(compute_cluster)

# Endpoint names need to be unique in a region, hence using timestamp to
# create unique endpoint name
timestamp = int(time.time())
endpoint_name = "speech-recognition-" + str(timestamp)

endpoint = BatchEndpoint(
    name=endpoint_name,
    description="Batch endpoint for "
    + foundation_model.name
    + ", for automatic-speech-recognition task",
)
ml_client.begin_create_or_update(endpoint).result()

deployment_name = "demo"

deployment = BatchDeployment(
    name=deployment_name,
    endpoint_name=endpoint_name,
    model=foundation_model.id,
    compute=compute_name,
    error_threshold=0,
    instance_count=1,
    logging_level="info",
    max_concurrency_per_instance=1,
    mini_batch_size=2,
    output_file_name="predictions.csv",
    retry_settings=BatchRetrySettings(max_retries=3, timeout=600),
)
ml_client.begin_create_or_update(deployment).result()

endpoint = ml_client.batch_endpoints.get(endpoint_name)

```

```

endpoint.defaults.deployment_name = deployment_name
ml_client.begin_create_or_update(endpoint).wait()

endpoint = ml_client.batch_endpoints.get(endpoint_name)
print(f"The default deployment is {endpoint.defaults.deployment_name}")
input = Input(path=batch_inputs_dir, type=AssetTypes.URI_FOLDER)

job = ml_client.batch_endpoints.invoke(
    endpoint_name=endpoint.name, input=input
)

ml_client.jobs.stream(job.name)

scoring_job = list(ml_client.jobs.list(parent_job_name=job.name))[0]

ml_client.jobs.download(
    name=scoring_job.name, download_path=batch_dir, output_name="score"
)

predictions_file = os.path.join(batch_dir, "named-outputs", "score",
"predictions.csv")

# Load the batch predictions file with no headers into a dataframe and set
your column names
score_df = pd.read_csv(
    predictions_file,
    header=None,
    names=["row_number_per_file", "prediction", "batch_input_file_name"],
)
score_df.head()

input_df = []
for file in input_files:
    input = pd.read_csv(os.path.join(batch_inputs_dir, file), index_col=0)
    input.reset_index(inplace=True)
    input["batch_input_file_name"] = file
    input.reset_index(names=["row_number_per_file"], inplace=True)
    input_df.append(input)
input_df = pd.concat(input_df)
input_df.set_index("index", inplace=True)
input_df = input_df.join(test_df.drop(columns=["audio", "language"]))

```

```

input_df.head()

df = pd.merge(
    input_df, score_df, how="inner", on=["row_number_per_file",
"batch_input_file_name"]
)

# Show the first few rows of the results
df.head(20)
ml_client.batch_endpoints.begin_delete(name=endpoint_name).result()
ml_client.compute.begin_delete(name=compute_name).result()

```

Link:

<https://github.com/srijonmandal1/schizophrenia-patient-data-analysis/blob/main/speech-analysis/AzureSpeechAnalysisNotebookV1.ipynb>

Sentiment Analysis, keyword Extraction and Summarization using OpenAI

Here goes a detailed example for performing all types of text analysis using OpenAI.

User needs to provide the sample text content.

```
=====
print("Review Content: {}".format(review_content))
# Use GPT-4 to classify the sentiment of the review content.
response1 = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "user",
            "content": "Classify the sentiment of the following review content following categories: \
categories: [Negative, Netural, Positive]\n\nreview content : " + review_content + \
"\n\nClassified sentiment:",
        },
    ],
)
classified_sentiment = response1.choices[0].message.content.replace(" ", "")
# print("Classified Sentiment of Review Content: {}".format(classified_sentiment))
print("Sentiment Classified")

# Use GPT-4 to find the different tones on the review content.
```

```

response2 = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "user",
            "content": "Based on the review content, find all the different tones with scores (out of 100). Show the scores in a json \
which has a structure [{}'tone': ..., 'score': ...},{'tone': ..., 'score': ...},... ] \
where 'tone' is the name of the key field and 'score' is the name of the value field:" \
+ review_content + "\n\nEmotional Tones:",
        },
    ],
)
emotional_tones = response2.choices[0].message.content.replace("\n","").replace(".","");
print("Emotional Tones Generated")

# Use GPT-4 to summarize the given content.
response3 = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "user",
            "content": "Based on the review content, find the key intents in maximum 5 keywords:" \
+ review_content + "\n\nIntent Keywords:",
        },
    ],
)
summarized_keywords =
response3.choices[0].message.content.replace("\n","").replace(".","");
print("Intent Keywords Generated")

# Append the insights result into a list.
review_content_list.append([review_content, classified_sentiment, emotional_tones,
summarized_keywords])

# Convert the list of insights into a Pandas dataframe.
review_content_df = pd.DataFrame(review_content_list, columns=['review_content',
'classified_sentiment', 'emotional_tones','intent_keywords'])
=====
```

```

[4]: # Primary functions to interact with AOAI GPT-3 to obtain insights.
review_content_list = []

for index, headers in df.iterrows():
    print(headers['Content'])
    review_content = str(headers['Content'])
    print("Review Content: {}.".format(review_content))
    # Use GPT-3 to classify the sentiment of the review content.
    response1 = client.chat.completions.create(
        model="gpt-4",
        messages=[
            {
                "role": "user",
                "content": "Classify the sentiment of the following review content following categories: \n categories: [Negative, Neutral, Positive]\nreview content : " + review_content + "\n\nClassified sentiment:"
            },
        ]
    )
    classified_sentiment = response1.choices[0].message.content.replace("\n", "")
    # print("Classified Sentiment of Review Content: {}".format(classified_sentiment))
    print("Sentiment Classified")

# Use AOAI GPT-4 to find the different tones on the review content.
response2 = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "user",
            "content": "Based on the review content, find all the different tones with scores (out of 100). Show the scores in a json \n which has a structure [{tone: ..., 'score': ...}, {tone: ..., 'score': ...}, ... ] \n where 'tone' is the name of the key field and 'score' is the name of the value field: \n + review_content = \"\n\nEmotional Tones\""
        },
    ],
)
emotional_tones = response2.choices[0].message.content.replace("\n", "").replace(" ", "")
print("Emotional Tones Generated")

# Use GPT-4 to summarize 3 keyword based on the review content.
response3 = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "user",
            "content": "Based on the review content, find the key intents in maximum 5 keywords:" +
            review_content + "\n\nIntent Keywords:"
        },
    ],
)

```

Audio Analysis using OpenAI Whisper Model

Following code shows how to transcribe the Audio data

=====

```
from openai import OpenAI
```

```
client = OpenAI(api_key=api_key_string)
file_path = "/Users/kaniska/Downloads/test.wav"
audio_file= open(file_path, "rb")
transcription = client.audio.transcriptions.create(
    model="whisper-1",
    file=audio_file
)
print(transcription.text)
```

```
import librosa
```

```
waveform, sample_rate = librosa.load(file_path, sr=None)
```

```
import matplotlib.pyplot as plt
```

```
time_axis = librosa.times_like(waveform, sr=sample_rate)
```

```
plt.figure(figsize=(10, 4))
plt.plot(time_axis, waveform)
```

```
plt.title('Waveform of Audio')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()

content_prompt = "Extract keywords from this text:\n\n"+str(transcription.text)

response = client.chat.completions.create(
    model="gpt-3.5-turbo-0125",
    response_format={"type":"json_object"},
    messages=[
        {"role": "system", "content": "You are a helpful assistant designed to output JSON."},
        {"role": "user", "content": content_prompt}
    ]
)

print(response.choices[0].message.content)

content_prompt = f"Please analyze the sentiment of the following text:{transcription.text}"

response = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "system",
            "content": "You will be provided with a text, and your task is to classify its sentiment as positive, neutral, or negative."
        },
        {
            "role": "user",
            "content": content_prompt
        }
    ],
    temperature=0.7,
    max_tokens=64,
    top_p=1
)

#sentiment = response.choices[0].content.strip().lower()

print(response)
=====
```

Translation using OpenAI Model

Following code shows how to use prompt and openai to analyze the transcription data generated from the above code

=====

```
import whisper
options = whisper.DecodingOptions(language= 'en', fp16=False)
model = whisper.load_model('medium', )
target_language = "hindi"
#content_prompt = f"Translate the following text into {target_language}: {transcription.text}\n",

translation = model.translate(transcription, target_language)
print(translation)

content_prompt = f"{transcription.text}"

response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[
        {
            "role": "system",
            "content": "You will be provided with a text, and your task is to translate the text into Hindi."
        },
        {
            "role": "user",
            "content": content_prompt
        }
    ],
    temperature=0.7,
    max_tokens=64,
    top_p=1
)

#translated_text = response.choices[0].content.strip().lower()

print(response)
```

```

File Edit View Run Kernel Tabs Settings Help
SpeechEmotionDetection.ipynb OpenAISpeechDocumentAr + Python 3 (pykernel)
[9]: content_prompt = f"Please analyze the sentiment of the following text:{transcription.text}"
response = client.chat.completions.create(
    model="gpt-4",
    messages=[
        {
            "role": "system",
            "content": "You will be provided with a text, and your task is to classify its sentiment as positive, neutral, or negative."
        },
        {
            "role": "user",
            "content": content_prompt
        }
    ],
    temperature=0.7,
    max_tokens=64,
    top_p=1
)
sentiment = response.choices[0].content.strip().lower()
print(response)

ChatCompletion(id='chatcmpl-91h1jNMFByp0OLHGTST35z1c1c7N', choices=[Choice(finish_reason='stop', index=0, logprobs=None, message=ChatCompletionMessage(content='Neutral', role='assistant', function_call=None, tool_calls=None))], created=1714242329, model='gpt-4-0013', object='chat.completion', system_fingerprint=None, usage=CompletionUsage(completion_tokens=1, prompt_tokens=89, total_tokens=90))

[ ]: !pip install -U openai-whisper
[ ]: import whisper
options = whisper.DecodingOptions(language='en', fp16=False)
model = whisper.load_model('medium')
target_language = "hindi"
content_prompt = f"Translate the following text into {target_language}: {transcription.text}\n"
translation = model.translate(transcription, target_language)
print(translation)

[13]: content_prompt = f"{transcription.text}"
response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[
        {
            "role": "system",
            "content": "You will be provided with a text, and your task is to translate the text into Hindi."
        },
        {
            "role": "user",
            "content": content_prompt
        }
    ],
    temperature=0.7,
    max_tokens=64,
    top_p=1
)
=====


```

Audio Waveform Analysis using HuggingFace Model

Following code shows how to use Huggingface library and model for Audio data analysis

=====

```
from transformers import pipeline
```

```
pipe = pipeline("audio-classification", model="MIT/ast-finetuned-audioset-10-10-0.4593")
```

```
results = pipe(waveform, sample_rate=sample_rate)
```

```
print(results)
```

```
pipe = pipeline("audio-classification",
model="ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition")
```

```
results = pipe(waveform, sample_rate=sample_rate)
```

```
print(results)
```

File Edit View Run Kernel Tabs Settings Help

Filter files by name

Name

- java2python
- OpenAISpeechDocumentAnaly...
- price-optimization (1).ipynb
- price-optimization-step-by-ste...
- SpeechEmotionDetection.ipynb

```
[ 1]: pip install openai
[ 2]: 
[ 3]: from openai import OpenAI
client = OpenAI(api_key='your_api_key_string')
file_path = "/Users/kaniska/Downloads/test.wav"
audio_file = open(file_path, "rb")
transcription = client.audio.transcriptions.create(
    model="whisper-1",
    file=audio_file
)
print(transcription.text)

I w-w-want a snack. I w-w-want to play. I am, I am r-r-r-running. I, I want a toy, toy. Take, take me. One more, one more.
```

```
[ 1]: pip install librosa
[ 2]: pip install matplotlib
[ 3]: import librosa
waveform, sample_rate = librosa.load(file_path, sr=None)
[ 4]: import matplotlib.pyplot as plt
time_axis = librosa.times_like(waveform, sr=sample_rate)
plt.figure(figsize=(10, 4))
plt.plot(time_axis, waveform)
plt.title('Waveform of Audio')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()
```

File Edit View Run Kernel Tabs Settings Help

Filter files by name

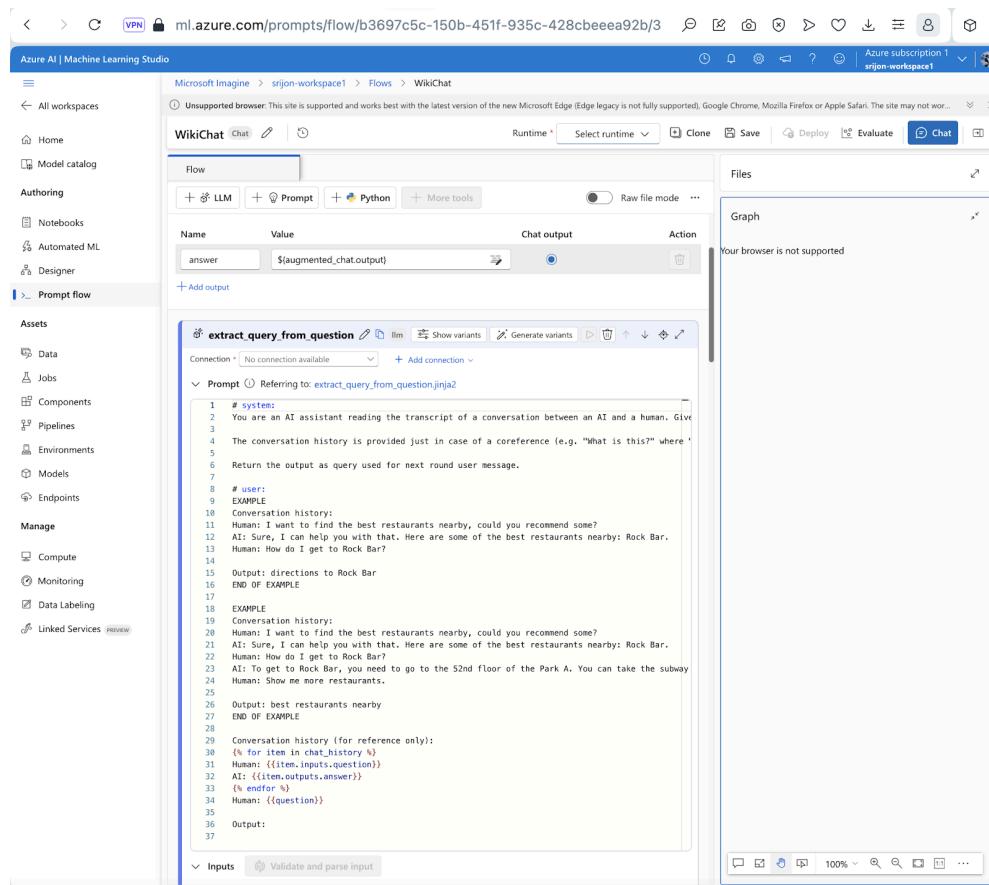
Name Last Modified

- java2python 2 years ago
- OpenAI... 4 hours ago
- price-opti... 1 month ago
- price-opti... 1 month ago
- SpeechEm... 4 hours ago

```
[ 1]: pip install transformers[torch]
[ 2]: python -c "from transformers import pipeline; print(pipeline('sentiment-analysis')('we love you'))"
[ 3]: import librosa
audio_path = "/Users/kaniska/Downloads/autism_3.wav"
waveform, sample_rate = librosa.load(audio_path, sr=None)
[ 4]: from transformers import pipeline
pipe = pipeline("audio-classification", model="MIT/ast-finetuned-audioset-10-10-0.4593")
results = pipe(waveform, sample_rate=sample_rate)
print(results)
[ 5]: pipe = pipeline("audio-classification", model="ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition")
results = pipe(waveform, sample_rate=sample_rate)
print(results)
```

/Users/kaniska/Library/jupyterlab-desktop/jlab_server/lib/python3.8/site-packages/transformers/configuration_utils.py:363: UserWarning: Passing `gradient_checkpointing` to a config initialization is deprecated and will be removed in v5 Transformers. Using `model.gradient_checkpointing_enable()` instead, or if you are using the `Trainer` API, pass `gradient_checkpointing=True` in your `TrainingArguments`.
warnings.warn(
Some weights of the model checkpoint at ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition were not used when initializing Wav2Vec2ForSequenceClassification: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.output.bias', 'classifier.output.weight', 'wav2vec2.encoder.pos_conv_embed.conv.weight_q', 'wav2vec2.encoder.pos_conv_embed.conv.weight_v']
- This IS expected if you are initializing Wav2Vec2ForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing Wav2Vec2ForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
Some weights of Wav2Vec2ForSequenceClassification were not initialized from the model checkpoint at ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition and are newly initialized: ['classifier.bias', 'classifier.weight', 'projector.bias', 'projector.weight', 'wav2vec2.encoder.pos_conv_embed.conv.parameters.weight.original1', 'wav2vec2.encoder.pos_conv_embed.conv.parameters.weight.original1']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[{'score': 0.1330530047416667, 'label': 'sad'}, {'score': 0.1317119308365448, 'label': 'fearful'}, {'score': 0.12580208480358124, 'label': 'neutral'}, {'score': 0.12434752285480499, 'label': 'calm'}, {'score': 0.12265881896618982, 'label': 'surprised'}]

Create a Chat Agent using Intelligent Prompts in Azure



Setup a Develop Environment in local Machine and Push Code to Azure

Provision a Virtual Machine and import Code from Github

- one can download the .pem file and login to the remote VM using SSH
- one can download the code from github into local machine and SCP to the remote VM

Run Application

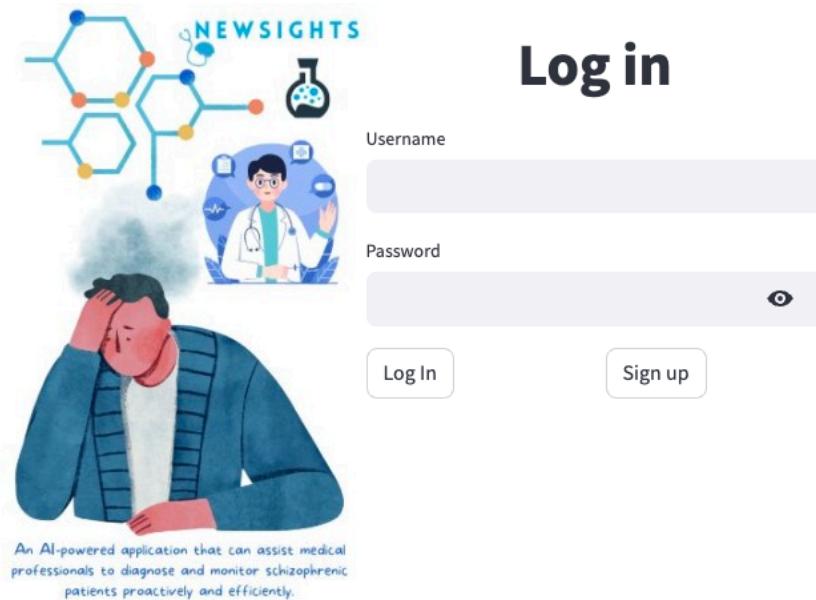
Run App

- Go to the app folder
 - cd streamlit-app
-
- Set OPENAI API Key
Export OPENAI_API_KEY in the VM
Set api_key_string in Prototype.py

- Run the Prototype
nohup python3 Prototype.py &
tail -f nohup.out
- View the App Use the localhost or IP Address assigned to the VM. For example, the below URL can be accessed by anyone. This Demo VM is running in Azure.

User Interaction

This is a prototype where a Lab User can login using admin credential.



The Lab User will be able to look up the overall patient status and insights in the Homepage. He will also be able to chat with a virtual agent which can search all the patient details and knowledgebase. This page also provides the overall patient stats.

Navigation

Navigator:

- Home
- KnowledgeBase
- Patient Diagnosis
- Ethical Guidelines
- Patient Registration



Welcome back!

This is a tool developed by the NewSights AI team for Microsoft ICJ 2024



What is Schizophrenia?

Schizophrenia is a complex and chronic mental health disorder marked by symptoms such as delusions, hallucinations, disorganized thinking and impaired social functioning, often co-occurs with depression, anxiety disorders, and substance abuse.

The symptoms not only significantly impact individual's functioning of daily life, but also pose a risk of harm to others.

Early intervention, comprehensive treatment approaches, and support from healthcare professionals, family, and community resources are essential in addressing the needs of individuals living with schizophrenia.

Please find more information in the information page.

Navigation

Navigator:

- Home
- KnowledgeBase
- Patient Diagnosis
- Ethical Guidelines
- Patient Registration



Global Prevalence

1% of the global population (~20M worldwide) affected
Source: World Health Organization



Total: 4 Patients

Add Patient

Select an existing patient:

Patient Information for Ander

Fullscreen

Fullname: *****

NewSights Analysis:

Age: 42

Gender: Male

Location: San Jose, CA

Medical History: History of Disorganized speech

Doctor Notes: (Date: Mar 19, 2024) Patient Ander, a 42-year-old male, has a history of depression. Current treatment regimen includes antidepressant medication and regular psychotherapy sessions. Patient's condition is stable with noticeable improvement in mood and overall well-being. Continued medication and therapy are recommended to maintain stability and prevent relapse.

Medical Tests Performed: MRI

Family Members with Condition: 2

[Update Profile](#)
[Perform Additional Diagnosis](#)
[Live Chat](#)

The Lab User will be able to ask Questions.

Chat with Virtual Agent

X

Agent: Hello Lab User! How can I assist you with your medical question today ?

Research User: I'm curious to learn more about patient David. What's the latest trend in his emotions ?

Agent: He is recently feeling remorseful about some past event. His depression level is high. Couple of days back the system detected some anomalous patterns in his voice tones.

Research User: Thanks for the update. I am here to help David. When is the next call scheduled with David ?

Agent: Next week, Monday he shall share another recording as per the schedule.
Anything else I can help with ?

Research User: Thank you. I will take the necessary action.



Close

The Lab user can select a patient and analyze specific Texts, Video and Speech Transcription data as shown below.

The Lab can perform extensive analysis on patient's data by leveraging the Generative AI functionalities explained in the previous sections. All the insights will be saved into a RAG database for Q&A and will also be saved in a Datastore for historical analysis and predictive analytics.

Analyze Textual Data

x

Navigation

Navigator:

- Home
- KnowledgeBase
- Patient Diagnosis
- Ethical Guidelines
- Patient Registration

**NEWSIGHTS**
AI Application to diagnose and monitor Schizophrenic patients

Patient Data Analysis for Ander

Choose Input Type:

- Text
- Audio
- Video

By selecting Text, Audio or Video data about patient, I authorize the release of the information including my personal heath record and diagnosis for medical treatment and research purpose.

Upload Text File

Cloud

Drag and drop file here

Limit 200MB per file • TXT, CSV, WAV, MP4, MOV, MPEG4

Browse files

Please Upload a Text File.

Analyze View Content

Uploaded Text

"Everything's just too much sometimes, you know? I get so mad because the voices won't stop yapping. It's like they're yelling right into my ears and I can't shut them off. Makes me so tired. I told the doctor, but I don't think he gets it. Not really. It feels like nobody understands.

And the people staring doesn't help. They look at me funny, whispering behind my back. It just makes me angrier. I try to smile, show them I'm okay, but inside I'm screaming. Why can't they just leave me alone?

The shadows are the worst. They sneak around, almost catching me, and then they're gone when I look again. It scares me. I hold onto my usual things, my routines, because without them, I feel like I'd just float away into some dark place.

Analyze Audio Data

Visual Insights

Navigation

Navigator:

- Home
- KnowledgeBase
- Patient Diagnosis
- Ethical Guidelines
- Patient Registration

Analyze Speech Text

Speech Tones

Tone Score	Score
82	Anger
79	Fear
75	Sadness
72	Frustration
70	Loneliness

Sentiment

20.8 % Positive
69.2 % Negative

MISUNDERSTAND SHADOWS
Voices
Routines Anger

Patient Summary

Audio
 Video

By selecting Text, Audio or Video data about patient, I authorize the release of the information including my personal health record and diagnosis for medical treatment and research purpose.

Navigation

Navigator:

- Home
- KnowledgeBase
- Patient Diagnosis
- Ethical Guidelines
- Patient Registration

Upload file

Drag and drop file here
Limit 200MB per file • WAV

Visual Insights

Analyze Speech Audio

Transcription

I w-w-want a snack. I w-w-want to play. I am, I am r-r-running.
I, I want a toy, toy. Take, take me. One more, one more.

Language Hindi Translation

मुझे एक नाश्ता चाहिए। मुझे खेलना है। मैं, मैं दोड़ रहा हूँ

Keywords Extracted

"snack" "play"
"running" "toy"
"take" "more"

Emotion Scale:

Emotion	Scale Value
Sad	Low
Fearful	Low
Calm	High
Surprised	Low

Analyze Video Data

×

● Video

By selecting Text, Audio or Video data about patient, I authorize the release of the information including my personal health record and diagnosis for medical treatment and research purpose.

Upload Video File

Drag and drop file here
Limit 200MB per file • TXT, CSV, WAV, MP4, MOV, MPEG4

Browse files

Please Upload a Video File.

Analyze

Visual Insights

Analyze Video Expressions



Emotional Distribution

Emotion	Percentage
Angry	55.0%
Confused	15.0%
Depressed	30.0%

Reference Codebase: <https://github.com/srijonmandal1/schizophrenia-patient-data-analysis>

Learn More

- Text Classification using Azure AI
 - <https://github.com/Azure/azureml-examples/blob/main/sdk/python/foundation-models/system/inference/text-classification/entailment-contradiction-batch.ipynb>
- Using Huggingface Model
 - <https://github.com/balakreshnan/Samples2021/blob/main/AzureML/hugginface1.md>