

whisper_evaluation

February 8, 2026

```
[8]: import os
from pydub import AudioSegment
import json

# -----
# Configuration
# -----
AUDIO_PATH = r"C:\Users\kupit\week 1\d4\lab2\PTT-20260207-WA0001.mp3"
WHISPER_PRICE_PER_MIN = 0.006 # USD per minute
OUTPUT_COST_JSON = r"C:\Users\kupit\week 1\d4\lab2\whisper_cost.json"

# -----
# Step 1: Get audio duration
# -----
if not os.path.exists(AUDIO_PATH):
    raise FileNotFoundError(f"Audio file not found: {AUDIO_PATH}")

audio = AudioSegment.from_file(AUDIO_PATH)

duration_ms = len(audio) #
duration_sec = duration_ms / 1000
duration_min_exact = duration_sec / 60 #

print(f"Audio duration: {duration_sec:.2f} seconds ({duration_min_exact:.4f}↵
↵minutes)")

# -----
# Step 2: Calculate exact cost
# -----
estimated_cost = duration_min_exact * WHISPER_PRICE_PER_MIN
print(f"Estimated Whisper API cost (@ $0.006/min): ${estimated_cost:.6f} USD")

# -----
# Step 3: Optional: multiple usage scenarios
# -----
scenarios_sec = [30, 60, 600, 3600, 7200] # : 30s, 1min, 10min, 1h, 2h
scenarios_costs = {}
```

```

print("\nCost estimates for different audio durations (seconds):")
for sec in scenarios_sec:
    cost = (sec / 60) * WHISPER_PRICE_PER_MIN
    scenarios_costs[sec] = round(cost, 6)
    print(f" {sec} sec -> ${cost:.6f} USD")

# -----
# Step 4: Save cost calculations to JSON
# -----
cost_data = {
    "audio_duration_sec": duration_sec,
    "audio_duration_min": duration_min_exact,
    "price_per_min": WHISPER_PRICE_PER_MIN,
    "estimated_cost": estimated_cost,
    "scenarios_costs": scenarios_costs
}

with open(OUTPUT_COST_JSON, "w", encoding="utf-8") as f:
    json.dump(cost_data, f, indent=2, ensure_ascii=False)

print(f"\n Cost calculation saved to JSON: {OUTPUT_COST_JSON}")

```

Audio duration: 50.58 seconds (0.8430 minutes)
 Estimated Whisper API cost (@ \$0.006/min): \$0.005058 USD

Cost estimates for different audio durations (seconds):

```

30 sec -> $0.003000 USD
60 sec -> $0.006000 USD
600 sec -> $0.060000 USD
3600 sec -> $0.360000 USD
7200 sec -> $0.720000 USD

```

Cost calculation saved to JSON: C:\Users\kupit\week
 1\d4\lab2\whisper_cost.json

```

[7]: import os
import json
from openai import OpenAI

# -----
# Configuration
# -----

# Path to the audio file (MP3)
AUDIO_PATH = r"C:\Users\kupit\week 1\d4\lab2\PTT-20260207-WA0001.mp3"

# Output transcription files

```

```

OUTPUT_TEXT_PATH = r"C:\Users\kupit\week 1\d4\lab2\transcription_full_audio.txt"
OUTPUT_JSON_PATH = r"C:\Users\kupit\week 1\d4\lab2\transcription_full_audio.
    ↪json"

# Initialize OpenAI client
client = OpenAI()

# -----
# Step 1: Verify audio file
# -----

if not os.path.exists(AUDIO_PATH):
    raise FileNotFoundError(f"Audio file not found: {AUDIO_PATH}")

file_size_mb = os.path.getsize(AUDIO_PATH) / (1024 * 1024)
file_extension = os.path.splitext(AUDIO_PATH)[1]

print("Audio file verification:")
print(f"- Path: {AUDIO_PATH}")
print(f"- File size: {file_size_mb:.2f} MB")
print(f"- File format: {file_extension}")
print("Audio file is accessible.\n")

# -----
# Step 2: Transcribe full audio
# -----

print("Starting transcription of full audio file...")

with open(AUDIO_PATH, "rb") as audio_file:
    transcription = client.audio.transcriptions.create(
        file=audio_file,
        model="whisper-1",
    )

transcription_text = transcription.text

# -----
# Step 3: Save transcription
# -----

# Save as text file
with open(OUTPUT_TEXT_PATH, "w", encoding="utf-8") as text_file:
    text_file.write(transcription_text)

# Save as JSON file
with open(OUTPUT_JSON_PATH, "w", encoding="utf-8") as json_file:

```

```

    json.dump({"text": transcription_text}, json_file, ensure_ascii=False,
    ↪indent=2)

# -----
# Output results
# -----

print("Transcription completed successfully.")
print(f"Text transcription saved to: {OUTPUT_TEXT_PATH}")
print(f"JSON transcription saved to: {OUTPUT_JSON_PATH}\n")

print("Transcription preview:")
print("-----")
print(transcription_text)

```

Audio file verification:

- Path: C:\Users\kupit\week 1\d4\lab2\PTT-20260207-WA0001.mp3
- File size: 0.60 MB
- File format: .mp3

Audio file is accessible.

Starting transcription of full audio file...

Transcription completed successfully.

Text transcription saved to: C:\Users\kupit\week
1\d4\lab2\transcription_full_audio.txt

JSON transcription saved to: C:\Users\kupit\week
1\d4\lab2\transcription_full_audio.json

Transcription preview:

When you exercise for a long time, you might hit a physical limit often called the wall. This isn't just your mind getting tired, it's a literal fuel crisis happening inside your cells. Your body carries two main types of energy, fat and glycogen. Think of fat as a giant, slow-burning log on a fire, while glycogen is like quick-burning kindling. Your body only stores a small amount of the fast-acting glycogen, and once it runs out, usually after about two hours of hard work, your system starts to panic. Your brain actually sends signals to shut your muscles down to save energy, making your legs feel like lead and your energy vanish instantly. To beat this, athletes train

```

[2]: import jiwer
import json
import os

```

```

# -----
# Configuration
# -----

```

```

WHISPER_TRANSCRIPT_PATH = r"C:\Users\kupit\week_1\d4\lab2\transcription_full_audio.txt"
GROUND_TRUTH_PATH = r"C:\Users\kupit\week 1\d4\lab2\transcription_full_audio - ground.txt"
OUTPUT_WER_PATH = r"C:\Users\kupit\week 1\d4\lab2\wer_results.json"

# -----
# Utility functions
# -----
def load_text(file_path):
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"File not found: {file_path}")
    with open(file_path, "r", encoding="utf-8") as f:
        return f.read()

# -----
# WER calculation logic
# -----
def calculate_wer(reference, hypothesis):
    transformation = jiwer.Compose([
        jiwer.ToLowerCase(),
        jiwer.RemovePunctuation(),
        jiwer.RemoveMultipleSpaces(),
        jiwer.Strip(),
    ])

    reference_clean = transformation(reference)
    hypothesis_clean = transformation(hypothesis)

    # WER
    wer = jiwer.wer(reference_clean, hypothesis_clean)

    # process_words
    measures = jiwer.process_words(reference_clean, hypothesis_clean)
    substitutions = measures.substitutions
    insertions = measures.insertions
    deletions = measures.deletions
    hits = measures.hits

    return {
        "wer": wer,
        "accuracy": 1 - wer,
        "substitutions": substitutions,
        "insertions": insertions,
        "deletions": deletions,
        "hits": hits,
        "reference_words": len(reference_clean.split()),
    }

```

```

        "hypothesis_words": len(hypothesis_clean.split()),
    }

def evaluate_whisper_accuracy(whisper_text, ground_truth_text):
    print("\n" + "=" * 50)
    print("EVALUATING WHISPER ACCURACY (WER)")
    print("=" * 50)

    wer_metrics = calculate_wer(ground_truth_text, whisper_text)

    print(f"\nWord Error Rate (WER): {wer_metrics['wer']:.4f}␣
↪({wer_metrics['wer']*100:.2f}%)"
    print(f"Accuracy: {wer_metrics['accuracy']:.4f}␣
↪({wer_metrics['accuracy']*100:.2f}%)"

    print("\nError Breakdown:")
    print(f"  Substitutions: {wer_metrics['substitutions']}")
    print(f"  Insertions:      {wer_metrics['insertions']}")
    print(f"  Deletions:          {wer_metrics['deletions']}")
    print(f"  Correct words: {wer_metrics['hits']}")

    print("\nWord Counts:")
    print(f"  Reference words: {wer_metrics['reference_words']}")
    print(f"  Hypothesis words: {wer_metrics['hypothesis_words']}")

    return wer_metrics

# -----
# Main execution
# -----
whisper_text = load_text(WHISPER_TRANSCRIPT_PATH)
ground_truth_text = load_text(GROUND_TRUTH_PATH)

wer_results = evaluate_whisper_accuracy(
    whisper_text=whisper_text,
    ground_truth_text=ground_truth_text
)

# Save results
with open(OUTPUT_WER_PATH, "w", encoding="utf-8") as f:
    json.dump(wer_results, f, indent=2)

print(f"\n WER evaluation saved to: {OUTPUT_WER_PATH}")

```

```

=====
EVALUATING WHISPER ACCURACY (WER)

```

=====

Word Error Rate (WER): 0.0168 (1.68%)

Accuracy: 0.9832 (98.32%)

Error Breakdown:

Substitutions: 2

Insertions: 0

Deletions: 0

Correct words: 117

Word Counts:

Reference words: 119

Hypothesis words: 119

WER evaluation saved to: C:\Users\kupit\week 1\d4\lab2\wer_results.json