Proyecto: Phonem Coach: Entrenamiento Personalizado de Fonemas con Análisis Acústico

Diplomado en Inteligencia Artificial y Ciencia de Datos

UNAM, Facultad de Ciencias

Alumna: Selene Martínez Ventura **Fecha de Entrega:** 25 de abril de 2025

The initial functions for the complete processing of this project will be listed below.

```
import os, json
import random, re
```

```
path_data = os.path.join("../data", "dictionary")
path_train = os.path.join("../data", "training")
```

1. Data dictionary phonemas by accent

The following lists of files by accent were created using the program src/phonema_by_accent.jl, using the database of CMU_ARTIC http://festvox.org/cmu_arctic/

```
if __name__ == "__main__":
    rutas_con_acentos = {
        "../data/phonemas/phonemas_cmu_us_awb_arctic.csv": "en-sc",
        "../data/phonemas/phonemas_cmu_us_bdl_arctic.csv": "en-us",
        "../data/phonemas/phonemas_cmu_us_clb_arctic.csv": "en-us",
        "../data/phonemas/phonemas_cmu_us_jmk_arctic.csv": "en-us",
        "../data/phonemas/phonemas_cmu_us_rms_arctic.csv": "en-us",
        "../data/phonemas/phonemas_cmu_us_slt_arctic.csv": "en-us",
    }
```

```
def generar_tokens_fonema_acento(rutas_archivos_con_acento):
    """
    Genera un diccionario de tokens (fonema: [palabras]) organizado por
acento.
    """
    tokens_por_acento = {}
    for ruta_archivo, acento in rutas_archivos_con_acento.items():
```

```
        tokens_por_fonema_temp = {}  # Diccionario temporal para acumular
sets de palabras
        try:
            with open(ruta_archivo, 'r', encoding='utf-8') as f:
                for linea in f:
                    partes = linea.strip().split(',')
                    if len(partes) >= 2:
                        palabra = partes[0].strip()
                        fonema = partes[1].strip()
                        fonema_limpio = re.sub(r'\d+|ˈ|ˌ', '', fonema)

                        if fonema_limpio not in tokens_por_fonema_temp:
                            tokens_por_fonema_temp[fonema_limpio] = set()
                        tokens_por_fonema_temp[fonema_limpio].add(palabra)

            # Convertir los sets temporales a listas ordenadas para el
acento actual
            tokens_por_acento[acento] = {
                fonema: sorted(list(palabras))
                for fonema, palabras in tokens_por_fonema_temp.items()
            }

        except FileNotFoundError:
            print(f"¡Error! No se encontró el archivo: {ruta_archivo}")

    return tokens_por_acento
```

```
tokens = generar_tokens_fonema_acento(rutas_con_acentos)
```

```
# Guardar los token en un archivo JSON
saving_path = os.path.join(path_data, 'tokens_by_accent.json')
with open(saving_path, 'w', encoding='utf-8') as f:
    json.dump(tokens, f, ensure_ascii=False, indent=4)

print("\nLos tokens por fonema: [palabras] se han guardado en:\n",
saving_path)
##
```

2. Fine tunning data

Se crean los datasets para realizar un ajuste fino al modelo de lenguaje preentrenado

```
RANGO_F1 = (350, 1000)
RANGO_F2 = (1000, 3000)
medias = {
    "æ": {"f1": [588, 669], "f2": [1952, 2349]},
    "ɪ": {"f1": [427, 483], "f2": [2034, 2365]},
```

```
    "ʌ": {"f1": [623, 753], "f2": [1200, 1426]},
}

medias_promedio = {}
```

```
path_token = os.path.join(path_data, "tokens_by_accent.json")
with open(path_token, 'r') as f:
    tokens_data = json.load(f)
```

```
vocales_info_base = {}
for accent_data in tokens_data.values():
    if isinstance(accent_data, dict):
        for fonema, palabras in accent_data.items():
            if fonema != "Fonema":
                palabras_filtradas = [palabra for palabra in palabras if
len(palabra) <= 5]
                if palabras_filtradas:
                    if fonema not in vocales_info_base:
                        vocales_info_base[fonema] = {"palabras":
palabras_filtradas, "std": [20, 40]}
                    else:
                        vocales_info_base[fonema]
["palabras"].extend(palabras_filtradas)

# Eliminar duplicados de las listas de palabras
for fonema in vocales_info_base:
    vocales_info_base[fonema]["palabras"] =
list(set(vocales_info_base[fonema]["palabras"]))
print("Claves en vocales_info_base:", vocales_info_base.keys())
```

```
Claves en vocales_info_base: dict_keys(['ʌ', 'ɪ', 'æ'])
```

```
# print("Primeras entradas de tokens_data:")
# for i, (key, value) in enumerate(tokens_data.items()):
#     if i < 5:
#         print(f"Clave: {key}, Valor: {value}")
#     else:
#         break
```

```
save_path_es = os.path.join(path_train, "data_es.txt")
```

```
save_path_en = os.path.join(path_train, "data_en.txt")
```

2.1 English version

```
import random

def generar_ejemplo_fine_tuning(vowel_key, medias):
    if vowel_key not in medias:
        print(f"Advertencia: La clave '{vowel_key}' no se encuentra en el
diccionario 'medias'.")
        return None

    f1_range_esperado = medias[vowel_key]["f1"]
    f2_range_esperado = medias[vowel_key]["f2"]
    info_base = vocales_info_base[vowel_key]
    palabra = random.choice(info_base["palabras"])

    # Generar F1 y F2 aleatorios dentro de los rangos del usuario
    f1_usuario = random.randint(350, 1000)
    f2_usuario = random.randint(1000, 3000)

    # Diagnóstico de F1
    if f1_usuario < f1_range_esperado[0]:
        comentario_f1 = f"The F1 value of {f1_usuario}Hz is too low for
/{vowel_key}/. Try opening your mouth a bit more. "
    elif f1_usuario > f1_range_esperado[1]:
        comentario_f1 = f"The F1 value of {f1_usuario}Hz is too high for
/{vowel_key}/. Try relaxing your jaw or opening your mouth slightly less. "
    else:
        comentario_f1 = f"The F1 value of {f1_usuario}Hz is within the
expected range for /{vowel_key}/. "

    # Diagnóstico de F2
    if f2_usuario < f2_range_esperado[0]:
        comentario_f2 = f"The F2 value of {f2_usuario}Hz is too low for
/{vowel_key}/. Try moving your tongue slightly forward. "
    elif f2_usuario > f2_range_esperado[1]:
        comentario_f2 = f"The F2 value of {f2_usuario}Hz is too high for
/{vowel_key}/. Try pulling your tongue back or relaxing it. "
    else:
        comentario_f2 = f"The F2 value of {f2_usuario}Hz is within the
expected range for /{vowel_key}/. "

    # Generar el prompt y la respuesta
    prompt = (
        f"The vowel /{vowel_key}/ was pronounced with F1={f1_usuario}Hz and
F2={f2_usuario}Hz. "
        f"The expected values are F1={f1_range_esperado[0]}-
{f1_range_esperado[1]}Hz "
        f"and F2={f2_range_esperado[0]}-{f2_range_esperado[1]}Hz\n"
```

```
            "Give feedback on the pronunciation:\n"
        )

        # Cierre motivador
        cierre = "Keep practicing the word and you'll improve quickly. 💪"

        completion = f"{comentario_f1}{comentario_f2}{cierre}"

        return {
            "prompt": prompt,
            "completion": " " + completion  # espacio inicial para el fine-
tuning
        }

# Ejemplo de generación del dataset para fine-tuning
num_ejemplos = 5000
with open(save_path_en, "w", encoding="utf-8") as f:
    vowels_from_json = list(vocales_info_base.keys())
    print(f"Vocales encontradas en vocales_info_base para generar ejemplos:
{vowels_from_json}")
    for i in range(num_ejemplos):
        if vowels_from_json:
            vowel = random.choice(vowels_from_json)
            ejemplo = generar_ejemplo_fine_tuning(vowel, medias)
            if ejemplo:
                f.write(f"{ejemplo['prompt']} {ejemplo['completion']}\n")
            else:
                print(f"La función devolvió None para la vocal: {vowel}")
        else:
            print("No se encontraron fonemas vocálicos en vocales_info_base
para generar ejemplos.")
            break

print(f"Se intentaron generar {num_ejemplos} ejemplos en {save_path_en}
para fine-tuning.")
```

```
Vocales encontradas en vocales_info_base para generar ejemplos: ['ʌ', 'ɪ',
'æ']
Se intentaron generar 5000 ejemplos en ../data/training/data_en.txt para
fine-tuning.
```

2. Fine tunning Spanish

```
def generar_ejemplo_fine_tuning(vowel_key, medias):
    if vowel_key not in medias:
        print(f"Advertencia: La clave '{vowel_key}' no se encuentra en el
diccionario 'medias'.")
        return None
```

```
    f1_range_esperado = medias[vowel_key]["f1"]
    f2_range_esperado = medias[vowel_key]["f2"]
    info_base = vocales_info_base[vowel_key]
    palabra = random.choice(info_base["palabras"])

    # Generar F1 y F2 aleatorios dentro de los rangos del usuario
    f1_usuario = random.randint(350, 1000)
    f2_usuario = random.randint(1000, 3000)

    # Diagnóstico de F1
    if f1_usuario < f1_range_esperado[0]:
        comentario_f1 = f"El valor de F1 de {f1_usuario}Hz es bajo para
/{vowel_key}/. Intenta abrir un poco más la boca. "
    elif f1_usuario > f1_range_esperado[1]:
        comentario_f1 = f"El valor de F1 de {f1_usuario}Hz es alto para
/{vowel_key}/. Intenta relajar la mandíbula o abrir un poco menos la boca.
"
    else:
        comentario_f1 = f"El valor de F1 de {f1_usuario}Hz está dentro del
rango esperado para /{vowel_key}/. "

    # Diagnóstico de F2
    if f2_usuario < f2_range_esperado[0]:
        comentario_f2 = f"El valor de F2 de {f2_usuario}Hz es bajo para
/{vowel_key}/. Intenta mover la lengua ligeramente hacia adelante. "
    elif f2_usuario > f2_range_esperado[1]:
        comentario_f2 = f"El valor de F2 de {f2_usuario}Hz es alto para
/{vowel_key}/. Intenta llevar la lengua hacia atrás o relajarla. "
    else:
        comentario_f2 = f"El valor de F2 de {f2_usuario}Hz está dentro del
rango esperado para /{vowel_key}/. "

    # Generar el prompt y la respuesta
    prompt = (
        f"La vocal {vowel_key} fue pronunciada con F1={f1_usuario}Hz and
F2={f2_usuario}Hz. "
        f"Los valores esperados son F1={f1_range_esperado[0]}-
{f1_range_esperado[1]}Hz "
        f"y F2={f2_range_esperado[0]}-{f2_range_esperado[1]}Hz \n"
        "Give feedback on the pronunciation:\n"
    )

    # Cierre motivador
    cierre = "Keep practicing the word and you'll improve quickly. 💪"

    completion = f"{comentario_f1}{comentario_f2}{cierre}"

    return {
        "prompt": prompt,
        "completion": " " + completion
    }

# Ejemplo de generación del dataset para fine-tuning
```

```
num_ejemplos = 5000
with open(save_path_es, "w", encoding="utf-8") as f:
    vowels_from_json = list(vocales_info_base.keys())
    print(f"Vocales encontradas en vocales_info_base para generar ejemplos:
{vowels_from_json}")
    for i in range(num_ejemplos):
        if vowels_from_json:
            vowel = random.choice(vowels_from_json)
            ejemplo = generar_ejemplo_fine_tuning(vowel, medias)
            if ejemplo:
                f.write(f"{ejemplo['prompt']} {ejemplo['completion']}\n")
            else:
                print(f"La función devolvió None para la vocal: {vowel}")
        else:
            print("No se encontraron fonemas vocálicos en vocales_info_base
para generar ejemplos.")
            break

print(f"Se intentaron generar {num_ejemplos} ejemplos en {save_path_es}
para fine-tuning.")
```

```
Vocales encontradas en vocales_info_base para generar ejemplos: ['ʌ', 'ɪ',
'æ']
Se intentaron generar 5000 ejemplos en ../data/training/data_es.txt para
fine-tuning.
```

3. Training model

3.1 Spanish version

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel, Trainer,
TrainingArguments, DataCollatorForLanguageModeling
from datasets import load_dataset
import shutil

#save_path_es = "/content/drive/MyDrive/ProyectoF/dataset_es.txt"

# Cargar tokenizer y modelo base
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
tokenizer.pad_token = tokenizer.eos_token
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Cargar y tokenizar dataset
dataset = load_dataset("text", data_files={"train": save_path_es})
def tokenize_function(examples):
    return tokenizer(examples["text"],
                     return_special_tokens_mask=True,
                     truncation=True,
```

```
                        max_length=256)

tokenized_datasets = dataset.map(tokenize_function, batched=True,
remove_columns=["text"])

# Data collator
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer, mlm=False, return_tensors="pt"
)

final_model_path = os.path.join("../data", "gpt2-pron-feedback_es")

# Argumentos de entrenamiento SIN checkpoints intermedios
training_args = TrainingArguments(
    output_dir=final_model_path,
    overwrite_output_dir=True,
    num_train_epochs=10,
    per_device_train_batch_size=32,
    gradient_accumulation_steps=1,
    save_strategy="no",  # No checkpoints
    logging_steps=10,
    logging_dir='./logs',
    learning_rate=3e-5,
    fp16=True,
    report_to="none",
)

# Entrenador
trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=tokenized_datasets["train"],
)

# Entrenar
trainer.train()

# Guardar modelo y tokenizer
model.save_pretrained(final_model_path)
tokenizer.save_pretrained(final_model_path)

# Comprimir para descargar
#shutil.make_archive(final_model_path, 'zip', final_model_path)
```

```
final_model_path_en = os.path.join("../data", "gpt2-pron-feedback_en")
final_model_path_es = os.path.join("../data", "gpt2-pron-feedback_es")
```

Trying model

Prueba del modelo

```
from transformers import pipeline
# Cargar modelo y tokenizer entrenados
model_path = final_model_path_es
tokenizer = final_model_path_es
generator = pipeline("text-generation", model=model_path,
tokenizer=tokenizer)
```

```
Device set to use cuda:0
```

```
phoneme = "ae"
input_word = "cat"
f1 = 3000
f2 = 2000
```

```
prompt = (
    f"La vocal /{phoneme}/ fue pronunciada con F1={int(f1)}Hz y F2=
{int(f2)}Hz. "
    f"La palabra practicada fue '{input_word}'. "
    f"Se detectó que el valor de F1 no está dentro del rango esperado. "
    "Brinda una sugerencia específica y clara para mejorar la pronunciación
de esta vocal en ese contexto.\nRespuesta:"
)

respuesta = generator(prompt, max_new_tokens=70, do_sample=True,
temperature=0.5)[0]["generated_text"]
print(respuesta)
```

```
La vocal /ae/ fue pronunciada con F1=3000Hz y F2=2000Hz. La palabra
practicada fue 'cat'. Se detectó que el valor de F1 no está dentro del
rango esperado. Brinda una sugerencia específica y clara para mejorar la
pronunciación de esta vocal en ese contexto.
Respuesta: ¡La apertura de tu boca es excelente, ¡Sigue así! 🎉 ∧ con
F1=1017Hz y F2=1023Hz., Se práctico la palabra 'lover'. ¡La apertura de tu
boca es excelente, ⁄
```