NOMBRE: Pablo Escobar

CARNE: 20936

FECHA: 29/09/2024

Comentario General: El objetivo de esta práctica es que exploren la arquitectura transformer que ya debieron haber leído en el paper original.

Cuando lleguen a la parte de entrenamiento y evaluación, procuren obtener el mejor resultado para WER. Cambien parametros, congelen capas convolucionales, etc. Dejen evidencia de todos sus intentos.

Responda las siguientes preguntas después de haber leído el paper original de Wav2Vec2(Se encuentra en: Materiales del profesor/papers/transformers y LLM): ¿Qué es Wav2Vec2 y qué problema intenta resolver? ¿Cómo se diferencia Wav2Vec2 de los métodos tradicionales de reconocimiento de voz? ¿Cuáles son los componentes principales de la arquitectura de Wav2Vec2? ¿Puedes explicar el proceso de entrenamiento de Wav2Vec2? ¿Cómo funciona el aprendizaje auto-supervisado en este contexto? ¿Qué papel juega la cuantización de características de audio en Wav2Vec2? ¿Cuáles son las ventajas y desventajas de usar Wav2Vec2 en comparación con otros modelos de reconocimiento de voz?

In [2]:
```
#Librerias a instalar, en caso de no estar instaladas

# !pip install datasets
# !pip install transformers
# !pip install jiwer
```

```
Requirement already satisfied: datasets in c:\users\dell\appdata\roaming\python\p
ython312\site-packages (3.0.1)
Requirement already satisfied: filelock in c:\users\dell\appdata\roaming\python\p
ython312\site-packages (from datasets) (3.15.4)
Requirement already satisfied: numpy>=1.17 in c:\users\dell\appdata\roaming\pytho
n\python312\site-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in c:\python312\lib\site-packages
(from datasets) (17.0.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in c:\python312\lib\site-packag
es (from datasets) (0.3.8)
Requirement already satisfied: pandas in c:\users\dell\appdata\roaming\python\pyt
hon312\site-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in c:\users\dell\appdata\roaming
\python\python312\site-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in c:\users\dell\appdata\roaming\pyth
on\python312\site-packages (from datasets) (4.66.4)
Requirement already satisfied: xxhash in c:\python312\lib\site-packages (from dat
asets) (3.5.0)
Requirement already satisfied: multiprocess in c:\users\dell\appdata\roaming\pyth
on\python312\site-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.6.1,>=2023.1.0 in c:\users\dell\appda
ta\roaming\python\python312\site-packages (from fsspec[http]<=2024.6.1,>=2023.1.0
->datasets) (2024.6.1)
Requirement already satisfied: aiohttp in c:\users\dell\appdata\roaming\python\py
thon312\site-packages (from datasets) (3.10.8)
Requirement already satisfied: huggingface-hub>=0.22.0 in c:\users\dell\appdata\r
oaming\python\python312\site-packages (from datasets) (0.24.5)
Requirement already satisfied: packaging in c:\users\dell\appdata\roaming\python
\python312\site-packages (from datasets) (24.1)
Requirement already satisfied: pyyaml>=5.1 in c:\users\dell\appdata\roaming\pytho
n\python312\site-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in c:\users\dell\appdata\r
oaming\python\python312\site-packages (from aiohttp->datasets) (2.4.2)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\dell\appdata\roaming
\python\python312\site-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in c:\users\dell\appdata\roaming\pyt
hon\python312\site-packages (from aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in c:\python312\lib\site-package
s (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\python312\lib\site-packa
ges (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.12.0 in c:\users\dell\appdata\roaming
\python\python312\site-packages (from aiohttp->datasets) (1.13.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\dell\appdat
a\roaming\python\python312\site-packages (from huggingface-hub>=0.22.0->datasets)
(4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\dell\appdata
\roaming\python\python312\site-packages (from requests>=2.32.2->datasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dell\appdata\roaming\pyth
on\python312\site-packages (from requests>=2.32.2->datasets) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\python312\lib\site-packag
es (from requests>=2.32.2->datasets) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dell\appdata\roamin
g\python\python312\site-packages (from requests>=2.32.2->datasets) (2024.7.4)
Requirement already satisfied: colorama in c:\users\dell\appdata\roaming\python\p
ython312\site-packages (from tqdm>=4.66.3->datasets) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\dell\appdata\ro
aming\python\python312\site-packages (from pandas->datasets) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\appdata\roaming\pyth
on\python312\site-packages (from pandas->datasets) (2024.1)
```

```
Requirement already satisfied: tzdata>=2022.7 in c:\users\dell\appdata\roaming\py
thon\python312\site-packages (from pandas->datasets) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\dell\appdata\roaming\python\p
ython312\site-packages (from python-dateutil>=2.8.2->pandas->datasets) (1.16.0)
[notice] A new release of pip is available: 24.1.2 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [5]:
```python
from datasets import load_dataset

# NO ENCONTRE EL DATASET SOLICITADO

timit = load_dataset("timit_asr", data_dir="timit")
```

```
Generating train split: 0 examples [00:00, ? examples/s]
Generating test split: 0 examples [00:00, ? examples/s]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[5], line 3
      1 from datasets import load_dataset
----> 3 timit = load_dataset("timit_asr", data_dir="timit")

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\load.py:2108, in l
oad_dataset(path, name, data_dir, data_files, split, cache_dir, features, downloa
d_config, download_mode, verification_mode, keep_in_memory, save_infos, revision,
token, streaming, num_proc, storage_options, trust_remote_code, **config_kwargs)
   2104 # Build dataset for splits
   2105 keep_in_memory = (
   2106     keep_in_memory if keep_in_memory is not None else is_small_dataset(bu
ilder_instance.info.dataset_size)
   2107 )
-> 2108 ds = builder_instance.as_dataset(split=split, verification_mode=verificat
ion_mode, in_memory=keep_in_memory)
   2109 if save_infos:
   2110     builder_instance._save_infos()

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\builder.py:1125, i
n DatasetBuilder.as_dataset(self, split, run_post_process, verification_mode, in_
memory)
   1122 verification_mode = VerificationMode(verification_mode or VerificationMod
e.BASIC_CHECKS)
   1124 # Create a dataset for each of the given splits
-> 1125 datasets = map_nested(
   1126     partial(
   1127         self._build_single_dataset,
   1128         run_post_process=run_post_process,
   1129         verification_mode=verification_mode,
   1130         in_memory=in_memory,
   1131     ),
   1132     split,
   1133     map_tuple=True,
   1134     disable_tqdm=True,
   1135 )
   1136 if isinstance(datasets, dict):
   1137     datasets = DatasetDict(datasets)

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\utils\py_utils.py:
512, in map_nested(function, data_struct, dict_only, map_list, map_tuple, map_num
py, num_proc, parallel_min_length, batched, batch_size, types, disable_tqdm, des
c)
    509         batch_size = max(len(iterable) // num_proc + int(len(iterable) %
num_proc > 0), 1)
    510     iterable = list(iter_batched(iterable, batch_size))
    511 mapped = [
--> 512     _single_map_nested((function, obj, batched, batch_size, types, None,
True, None))
    513     for obj in hf_tqdm(iterable, disable=disable_tqdm, desc=desc)
    514 ]
    515 if batched:
    516     mapped = [mapped_item for mapped_batch in mapped for mapped_item in m
apped_batch]

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\utils\py_utils.py:
373, in _single_map_nested(args)
    371         return function([data_struct])[0]
    372     else:
```

```
--> 373         return function(data_struct)
    374 if (
    375     batched
    376     and not isinstance(data_struct, dict)
    377     and isinstance(data_struct, types)
    378     and all(not isinstance(v, (dict, types)) for v in data_struct)
    379 ):
    380     return [mapped_item for batch in iter_batched(data_struct, batch_siz
e) for mapped_item in function(batch)]

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\builder.py:1155, i
n DatasetBuilder._build_single_dataset(self, split, run_post_process, verificatio
n_mode, in_memory)
   1152     split = Split(split)
   1154 # Build base dataset
-> 1155 ds = self._as_dataset(
   1156     split=split,
   1157     in_memory=in_memory,
   1158 )
   1159 if run_post_process:
   1160     for resource_file_name in self._post_processing_resources(split).valu
es():

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\builder.py:1229, i
n DatasetBuilder._as_dataset(self, split, in_memory)
   1227 if self._check_legacy_cache():
   1228     dataset_name = self.name
-> 1229 dataset_kwargs = ArrowReader(cache_dir, self.info).read(
   1230     name=dataset_name,
   1231     instructions=split,
   1232     split_infos=self.info.splits.values(),
   1233     in_memory=in_memory,
   1234 )
   1235 fingerprint = self._get_dataset_fingerprint(split)
   1236 return Dataset(fingerprint=fingerprint, **dataset_kwargs)

File ~\AppData\Roaming\Python\Python312\site-packages\datasets\arrow_reader.py:25
1, in BaseReader.read(self, name, instructions, split_infos, in_memory)
    249 if not files:
    250     msg = f'Instruction "{instructions}" corresponds to no data!'
--> 251     raise ValueError(msg)
    252 return self.read_files(files=files, original_instructions=instructions, i
n_memory=in_memory)

ValueError: Instruction "train" corresponds to no data!
```

In [6]:
```
#Elimine las columnas innecesarias(para este caso didactico). Solo necesita "tex


def remove_unnecessary_columns(dataset):
    columns_to_keep = ["text", "audio", "file", "id"]
    columns_to_remove = [col for col in dataset.column_names if col not in colum

    # Eliminar las columnas innecesarias
    dataset = dataset.remove_columns(columns_to_remove)
    return dataset
```

In [7]:
```
from datasets import ClassLabel
import random
```

```python
import pandas as pd

# Defina la función que recibe el dataset y un entero n
def show_random_texts(dataset, n):
    random_indices = random.sample(range(len(dataset)), n)

    sample_data = dataset.select(random_indices)

    df = pd.DataFrame({
        "id": sample_data["id"],
        "text": sample_data["text"],
        "audio": sample_data["audio"],
        "file": sample_data["file"]
    })

    return df
```

In [8]:
```python
import re
#Limpie los textos.
import unicodedata

# Función para limpiar los textos
def clean_text(text):
    text = text.lower()
    text = unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode(
    text = re.sub(r'[^a-z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

def clean_dataset_texts(dataset):
    dataset = dataset.map(lambda x: {"text": clean_text(x["text"])})
    return dataset
```

In [9]:
```python
#Defina una funcion para construir el vocabulario. Esto es todos los caracteres
#Ejemplo: {a: 3, b:5, c:7}
#Luego, cambie el espacio " " por "|". Esto es una buena practica, para tener un
#Al vocabulario debe agregar "[UNK]" y "[PAD]". Uno es para desconocidos, otro e
#la longitud del vocabulario.
#Guarde el vocabulario en un json: vocab.json



import json
from collections import Counter

def build_vocabulary(dataset):
    vocab_counter = Counter()
    for text in dataset["text"]:
        vocab_counter.update(text)
    if " " in vocab_counter:
        vocab_counter["|"] = vocab_counter.pop(" ")
    return vocab_counter

def save_vocabulary(vocab_counter, filepath):
    vocab_size = len(vocab_counter)
    vocab_counter["[UNK]"] = vocab_size
    vocab_counter["[PAD]"] = vocab_size + 1
    with open(filepath, "w") as vocab_file:
```

```
        json.dump(vocab_counter, vocab_file, ensure_ascii=False, indent=4)
    print(f"Vocabulario guardado en: {filepath}")
```

In [ ]:
```
from transformers import Wav2Vec2CTCTokenizer
#Use el vocabulario anterior para tokenizar con Wav2Vec2TCTokenizer


tokenizer = Wav2Vec2CTCTokenizer(
    "vocab.json",
    unk_token="[UNK]",
    pad_token="[PAD]",
    word_delimiter_token="|"
)
```

In [ ]:
```
from transformers import Wav2Vec2FeatureExtractor
#Defina el pipeline de extraccion de caracteristicas


feature_extractor = Wav2Vec2FeatureExtractor(
    feature_size=1,
    sampling_rate=16000,
    padding=True,
    do_normalize=True,
    return_attention_mask=True
)
```

In [ ]:
```
from transformers import Wav2Vec2Processor

processor = Wav2Vec2Processor(
    feature_extractor=feature_extractor,
    tokenizer=tokenizer
)
```

In [ ]:
```
#Antes de empezar, evalue que todo esta bien: escuche un par de audios y vea si
```

In [ ]:
```
#Use la siguiente funcion para crear los batches en el formato que el modelo nec
def prepare_dataset(batch):
    audio = batch["audio"]

    # batched output is "un-batched" to ensure mapping is correct
    batch["input_values"] = processor(audio["array"], sampling_rate=audio["sampl
    batch["input_length"] = len(batch["input_values"])

    with processor.as_target_processor():
        batch["labels"] = processor(batch["text"]).input_ids
    return batch
```

In [ ]:
```
#Training enviroment: use este como guía, puede que varíe respecto a como nombra
import torch

from dataclasses import dataclass, field
from typing import Any, Dict, List, Optional, Union

@dataclass
class DataCollatorCTCWithPadding:

    processor: Wav2Vec2Processor
    padding: Union[bool, str] = True
```

```
    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]
        # split inputs and labels since they have to be of different lenghts and
        # different padding methods
        input_features = [{"input_values": feature["input_values"]} for feature
        label_features = [{"input_ids": feature["labels"]} for feature in featur

        batch = self.processor.pad(
            input_features,
            padding=self.padding,
            return_tensors="pt",
        )
        with self.processor.as_target_processor():
            labels_batch = self.processor.pad(
                label_features,
                padding=self.padding,
                return_tensors="pt",
            )

        # replace padding with -100 to ignore loss correctly
        labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_ma

        batch["labels"] = labels

        return batch
```

In [ ]:
```
data_collator = DataCollatorCTCWithPadding(processor=processor, padding=True)
```

In [ ]:
```
#Metrica de rendimiento WER
wer_metric = load_metric("wer")
def compute_metrics(pred):
    pred_logits = pred.predictions
    pred_ids = np.argmax(pred_logits, axis=-1)

    pred.label_ids[pred.label_ids == -100] = processor.tokenizer.pad_token_id

    pred_str = processor.batch_decode(pred_ids)
    label_str = processor.batch_decode(pred.label_ids, group_tokens=False)

    wer = wer_metric.compute(predictions=pred_str, references=label_str)

    return {"wer": wer}
```

In [ ]:
```
from transformers import Wav2Vec2ForCTC

model = Wav2Vec2ForCTC.from_pretrained(
    "facebook/wav2vec2-base",
    ctc_loss_reduction="mean",
    pad_token_id=processor.tokenizer.pad_token_id,
)
```

In [ ]:
```
#Parametros de entrenamiento segun la documentacion

from transformers import TrainingArguments

training_args = TrainingArguments(
  output_dir=repo_name,
  group_by_length=True,
  per_device_train_batch_size=8,
```

```
  evaluation_strategy="steps",
  num_train_epochs=30,
  fp16=True,
  gradient_checkpointing=True,
  save_steps=500,
  eval_steps=500,
  logging_steps=500,
  learning_rate=1e-4,
  weight_decay=0.005,
  warmup_steps=1000,
  save_total_limit=2,
)
```

In [ ]:

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    data_collator=data_collator,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=timit["train"],
    eval_dataset=timit["test"],
    tokenizer=processor.feature_extractor,
)
```