تمرین چهارم درس پردازش زبان و گفتار

استاد درس: دکتر حمیدرضا برادران کاشانی

دستیاران آموزشی: آیین کوپایی ـ هاجر مظاهری

تاریخ بارگذاری تمرین: ۱۴۰۳/۰۳/۲۳

تاریخ تحویل تمرین: ۱۴۰۳/۰۴/۱۲

دانشگاه اصفهان

هدف تمرین:

- آشنایی با آموزش نیمه نظارتی (semi-supervised training)
- آشنایی با مدل wav2vec 2.0 از پیش آموزش دیده برای تشخیص خودکار گفتار
- تنظیم دقیق مدل wav2vec 2.0 با داده‌های گفتاری برچسب‌گذاری شده بسیار کم
- ساخت یک سیستم بازشناسی گفتار فارسی

مقاله خوانی:

با توجه به مقاله [3] و[4] به سوالات زیر پاسخ دهید:

- معماری wav2vec 2.0 و فرآیند پیش آموزش را با جزئیات توضیح دهید.
- تفاوت مدل wav2vec2.0 و wav2vec xlsr-53 در چیست؟
- رمزگشایی در مدل wav2vec2.0 با چه الگوریتمی انجام می‌شود؟ روش را توضیح دهید.
- برای بهبود نتایج بدست آمده چه روش یا تکنیکی را پیشنهاد می‌کنید.

مراحل تمرین:

۱- دسترسی به مجموعه داده

برای تنظیم دقیق مدل wav2vec2 از مجموعه داده موزیلا (common voice) استفاده می‌کنیم. جدول ۱ مشخصات مجموعه داده را نشان می دهد.

جدول ۱: مجموعه داده

Dataset	Version	language
common_voice	۶.۱	Persian

روش دسترسی به مجموعه داده موزیلا:

- روش آفلاین: به سایت common_voice به آدرس[1] رفته و در قسمت زبان، زبان فارسی (Persion) را انتخاب کنید و با پذیرفتن تعهدات و ثبت ایمیل خود مجموعه داده را دانلود کنید.

- روش آنلاین: با استفاده از کتابخانه dataset به صورت زیر مجموعه داده را از مخزن Hugging Face دانلود کنید.

common_voice_train = load_dataset("mozilla-foundation/common_voice_6_1", "fa", split="train")

common_voice_ test = load_dataset("mozilla-foundation/common_voice_6_1", "fa", split=" test ")

نکته: به سایت [2] رفته و Agree and access را تایید کنید.

۲- حذف اطلاعات اضافی از مجموعه داده

در مجموعه‌های داده ASR ما فقط به متن هدف(sentence) برای هر آرایه صوتی (audio) و مسیر فایل صوتی (path) نیاز داریم. اما مجموعه داده موزیلا اطلاعات بسیار بیشتری را ارائه می‌دهد که در بازشناسی گفتار مورد نظر استفاده نمی‌شود. پس می‌توان ستون‌های زیر را با استفاده از دستور remove_columns از مجموعه داده حذف کرد:

down_votes, gender, locale, segment, up_votes , accent, age, client_id

۳- پیش پردازش داده (نرمال‌سازی و حذف کارکترهای خاص از جملات)

پیش‌پردازش داده‌ها یک مرحله بسیار مهم قبل از آموزش مدل است. به عنوان مثال، نباید مدل بین آ و ا تفاوت قائل شود. تفاوت بین ا و آ به هیچ وجه به "صدای" حرف بستگی ندارد، بلکه بیشتر به قوانین دستوری بستگی دارد . بنابراین منطقی است که تفاوت بین حروف بزرگ و غیر بزرگ را حذف کنیم.

در جملات مجموعه داده مشاهده می‌کنید که از علائم نگارشی و کارکترهای خاص مانند (نقطه، علامت سوال،...) استفاده شده است. این علایم نمی‌توانند با صدای آکوستیک نمایش داده شوند و بهتر است این کارکترها قبل از آموزش مدل حذف شوند.

در این مرحله می‌توانید با استفاده از کتابخانه hazm.Normalizer نرمال‌سازی جملات فارسی را انجام دهید. نرمسازی حروف فارسی را می‌توانید به صورت زیر انجام دهید:

```
chars_to_mapping = {
    'ك': 'ک', 'ۀ': 'ه', 'ب': 'ب', 'ز': 'ز', 'ذ': 'ذ', 'ش': 'ش', 'س': 'س', 'ى': 'ی',
    'ي': 'ی', 'أ': 'ا', 'ؤ': 'و', 'ے': 'ی', 'ۀ': 'ه', 'پ': 'پ', 'ک': 'ک', 'ی': 'ی',
    'ﻟ': 'ا', 'ﺑ': 'ب', 'ﺗ': 'ت', 'ﺧ': 'خ', 'د': 'د', 'ﺲ': 'س', 'ﺿ': 'ض', 'ﻋ': 'ع',
    'ﻠ': 'ل', 'ﻣ': 'م', 'ﻤ': 'م', 'ﻪ': 'ه', 'و': 'و', 'ا': 'ا', 'ة': 'ه',
    'ﻴ': 'ی', 'ﻳ': 'ی', 'ﺒ': 'ب', 'ت': 'ت', 'ﺪ': 'د', 'ر': 'ر', 'ﺳ': 'س', 'ش': 'ش',
    'ﺷ': 'ش', 'ﻋ': 'ع', 'ﻫ': 'م', 'ن': 'ن', 'ﻨ': 'ن', 'و': 'و', 'ر': 'ر', 'گ': 'گ',

    # 'ها': ' ها', 'ئ': 'ی',
    '۱۴ام': '۱۴ ام',

    'a': ' ای ', 'b': ' بی ', 'c': ' سی ', 'd': ' دی ', 'e': ' ایی ', 'f': ' اف ',
    'g': ' جی ', 'h': ' اچ ', 'i': ' آی ', 'j': ' جی ', 'k': ' کی ', 'l': ' ال ',
    'm': ' ام ', 'n': ' ان ', 'o': ' او ', 'p': ' پی ', 'q': ' کیو ', 'r': ' آر ',
    's': ' اس ', 't': ' تی ', 'u': ' یو ', 'v': ' وی ', 'w': ' دبلیو ', 'x': ' اکس ',
    'y': ' وای ', 'z': ' زد ',
    '\u200c': ' ', '\u200d': ' ', '\u200e': ' ', '\u200f': ' ', '\ufeff': ' ',
}
```

لیست کارکترهای حذفی می‌تواند به صورت زیر باشد:

```
Chars_to_ignore                                                    =                                                    [
  ', \'?', ؟     \',', \'?', \'.', \'!', \'-', \';', \':', '\'\', \'%', \'\', '\'', '\'', '\'','#', \'!', \
\'«\', \'»\', \'،', \'(', \')', \'؛', \'\',\',\'', \',', \'?',  \'.', \'!', \'-',
\';',\':','\'',''\', \'%', \'\',\'=\',  \'-', \'…', \'_', \'\', '\'', '„','ā', 'š',]
```

۴- ساخت دیکشنری

تمام حروف متمایز در داده‌های آموزش و تست را استخراج می‌کنیم و در دیکشنری ذخیره می‌کنیم.

نکته:

- تعداد حروف متمایز در مجموعه داده موزیلا فارسی ۳۵ است .
- یک نشانه unknown باید اضافه شود تا مدل بعداً بتواند با کاراکترهایی که در مجموعه آموزشی موزیلا با آنها مواجه نشده‌اند برخورد کند.
- توکن <s></s> نیز برای مشخص کردن ابتدا و انتهای جملات اضافه می شود.
- برای مشخص شدن هر توکن" "(خالی) به جای آن یک کاراکتر "|" قرار داده می‌شود. (فاصله بین کلمات)

۵- توکنایز کردن و استخراج ویژگی‌ها

از کلاس Wav2Vec2CTCTokenizer برای توکنایز کردن جملات استفاده کنید.

توجه داشته باشید که مجموعه داده موزیلا با فرکانس فرکانس ۴۸ کیلوهرتز نمونه‌برداری شده است و قبل از آموزش باید فرکانس نمونه برداری به ۱۶ کیلوهرتز کاهش یابد.(راهنمایی: فایل ها صوتی با فرمت mp3 هستند. آنها را بااستفاده از کتابخانه torchaudio یا librosa خوانده و سپس با استفاده از همین کتابخانه‌ها و تابع map، فرکانس ۴۸ کیلوهرتز را به ۱۶ کیلوهرتز نگاشت دهید.)

استخراج ویژگی Wav2Vec2FeatureExtractor را با استفاده از کتابخانه transformers فراخوانی کنید.

استخراج ویژگی‌ها به پارامترهای زیر نیاز دارد:

- Feature size: مدل‌های گفتار، دنباله‌ای از بردارهای ویژگی را به عنوان ورودی می‌گیرند در حالی که طول این دنباله به متفاوت است. در مورد Wav2Vec2 ، اندازه ویژگی ۱ است زیرا مدل بر روی سیگنال گفتار خام آموزش داده شده است.
- Sample rate: نرخ نمونه برداری که مدل بر روی آن آموزش می‌بیند. (۱۶ کیلوهرتز)
- Do_normalize: بهتر است مقدار این گزینه True باشد تا نرمالسازی انجام شود.
- Return_attention_mask: مقدار این گزینه True است زیرا به طور کلی، مدل های Wav2Vec2 باید از ماسک توجه استفاده کنند.

۶- ایجاد یک Processor

برای ساده‌سازی استفاده از استخراج‌کننده ویژگی و توکنایزر، می‌توانیم هر دو را در یک کلاس Wav2Vec2Processor قرار دهیم. در زمان اجرا فقط به یک مدل وprocessor نیاز داریم. Wav2Vec2Processor را می‌توان از کتابخانه transformers فراخوانی کرد.

۷- فیلتر زمانی

برای آموزش مدل از فایل های صوتی با طول ۴ تا ۶ ثانیه استفاده کنید و برای تست از فایل‌های با طول کمتر ۱۵ ثانیه استفاده کنید.

(تعداد فایل های آموزش و تست را گزارش کنید.)

۸- Data Collator و pading

جمع‌آوری داده را تعریف کنید جمع‌آوری داده برای مدل‌های گفتاری منحصربه‌فرد است زیرا ویژگی‌های ورودی و برچسب‌ها به‌صورت مستقل بررسی می‌شوند. ویژگی‌های ورودی باید توسط استخراج‌کننده ویژگی و برچسب‌ها توسط توکنایزر مدیریت شوند. همچنین برخلاف اکثر مدل‌هایNLP در مدل Wav2Vec2 طول ورودی بسیار بیشتری از طول خروجی است. به عنوان مثال، یک نمونه با طول ورودی ۵۰۰۰۰ دارای طول خروجی حدود ۱۰۰ است. بنابراین آموزش مدل Wav2Vec2 به padding ویژه نیاز دارد.

۹- معیار ارزیابی

برای ارزیابی سیستم بازشناسی گفتار از معیار نرخ خطای کلمه(WER) استفاده کنید.

۱۰- بارگیری مدل

مدل facebook/wav2vec2-large-xlsr-53 را با استفاده Wav2Vec2ForCTC بارگیری کنید.

۱۱- تنظم مدل wav2vec2 با مجموعه داده موزیلا

ابتدا بخش استخراج کننده ویژگی را فریز کنید و سپس با توجه به جدول ۲ زیر مدل را آموزش دهید.

جدول۲:تنظیمات اجرا

num_train_epochs	5
fp16	True
attention_dropout	0.1
learning_rate	1e-4
warmup_steps	1000

۱۲- پکیج‌های مورد نیاز برای تنظیم مدل wav2vec2

Datasets >= 1.18.3
Transformers == 4.11.3
Librosa
Jiwer
Torchaudio
Hazm
Num2fawords

۱۳- تنظیم batch_size

با توجه به جدول زیر می‌توانید اندازه batch_size را تنظیم کنید.

جدول۳: تنظیمات batch_size

سخت افزار	Batch-size
T4 16G (Colab)	Batch-size<=12
RTX 8 G	Batch-size<=6

نکات تحویل

۱- پاسخ خود را در پوشه ای به اسم NLP_NAME_FAMILY_HW4 و در قالب zip بارگذاری نمایید.

۲- این پوشه باید حاوی موارد زیر باشد:

- کد نوشته شده در قالب یک فایل jupyter notebook
- تشریح و تحلیل نتایج بدست آمده از نظر شما
- آپلود مدل در یک فضای ابری یا مخزن هاینگ فیس و ارسال لینک
- پاسخ سوالات مقاله خوانی
- گزارش نتایج بدست آمده از تنظیم دقیق مدل wav2vec مطابق با جدول ۴

جدول۴: نتایج اجرا

Batch Size
Num Epoch
تعداد فایل‌های مورد استفاده برای آموزش مدل
تعداد فایل‌های مورد استفاده برای تست مدل
مدت زمان آموزش
تعداد پارامترهای مدل
تعداد پارامترهای فعال در زمان آموزش مدل
WER (Word Error Rate)

۳- لازم به ذکر است که رعایت قوانین نگارشی حائز اهمیت است.

مراجع:

[1] https://commonvoice.mozilla.org/en/datasets
[2] https://huggingface.co/datasets/mozilla-foundation/common_voice_6_1
[3] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, *33*, 12449-12460.

[4] Conneau, A., Baevski, A., Collobert, R., Mohamed, A., & Auli, M. (2020). Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*.