

تشریح و تحلیل نتایج بدست آمده



دانشکده مهندسی کامپیوتر

تکلیف چهارم درس مبانی NLP

استاد درس: دکتر برادران

دستیاران درس:

آئین کوپایی، هاجر مظاهری

سیاوش امیرحاجلو

993613007

تیر 1403

مقدمه

این گزارش تحلیلی جامع از فرآیند تنظیم دقیق مدل wav2vec-large-xlsr-53 ارائه می‌کند و آن را برای تشخیص گفتار فارسی تطبیق می‌دهد. مدل wav2vec2 که در اصل توسط فیس‌بوک AI توسعه داده شد، عملکرد قابل توجهی در وظایف تشخیص گفتار چند زبانه و چند زبانه از خود نشان داده است. هدف پروژه ما استفاده از این معماری قدرتمند برای بهبود قابلیت‌های تشخیص خودکار گفتار (ASR) برای زبان فارسی است.

فرآیند تنظیم دقیق از مجموعه داده‌های Common Voice برای فارسی، مجموعه ای از ضبط و رونویسی‌های صوتی در دسترس عموم استفاده می‌کند. با تطبیق مدل از پیش آموزش دیده wav2vec-large-xlsr-53 با این زبان خاص، هدف ما ایجاد یک سیستم ASR با کارایی بالا متناسب با الگوهای گفتاری و آوایی فارسی است.

روش شناسی ما شامل چندین مرحله کلیدی است:

1. آماده سازی داده‌ها و پیش پردازش مجموعه داده فارسی Common Voice
2. پیکربندی و بهینه سازی معماری مدل wav2vec2
3. پیاده سازی خط لوله آموزشی سفارشی با استفاده از کتابخانه Hugging Face Transformers
4. ارزیابی عملکرد مدل با استفاده از متریک Word Error Rate (WER).

این گزارش به جزئیات جنبه‌های فنی رویکرد ما، از جمله چالش‌های خاص در پردازش متن و داده‌های صوتی فارسی، تغییرات ایجاد شده در معماری مدل، و استراتژی‌های آموزشی بکار گرفته شده برای دستیابی به نتایج بهینه، پرداخته است. ما همچنین تجزیه و تحلیل کاملی از عملکرد مدل ارائه خواهیم داد.

مجموعه داده

برای این پروژه از زیرمجموعه فارسی مجموعه داده common voice نسخه 6.1 که توسط شرکت موزیلا توسعه یافته است، استفاده کردیم.

ویژگی‌های کلیدی مجموعه داده

1. زبان: مجموعه داده به طور خاص بر زبان فارسی تمرکز دارد.
2. تقسیم مجموعه داده ها: مجموعه داده به سه زیر مجموعه اصلی تقسیم شده
 - مجموعه آموزشی: ترکیبی از تقسیم‌های اصلی "Train" و "Validation".
 - مجموعه تست: تقسیم "Test" اصلی
3. قالب داده: هر ورودی در مجموعه داده شامل موارد زیر است
 - یک فایل صوتی (اصلی با فرمت MP3، تبدیل به WAV)
 - متن مربوطه
4. پیش پردازش
 - صدا: فایل‌های صوتی اصلی از 48 کیلوهرتز به 16 کیلوهرتز نمونه‌برداری شدند تا با نیازهای ورودی مدل wav2vec2 مطابقت داشته باشند.

```
common_voice_train = common_voice_train.cast_column("audio", Audio(sampling_rate=16_000))
common_voice_test = common_voice_test.cast_column("audio", Audio(sampling_rate=16_000))
```

قطعه کد 1: تغییر فرکانس نمونه برداری

- متن: عادی سازی متن گسترده انجام شد، از جمله:

- استاندارد کردن کاراکترهای فارسی

- نگاشت کاراکترهای غیر فارسی به معادل فارسی آن ها

- حذف کاراکترهای خاص و نشانه ها

- عادی سازی فضای خالی

```
def preprocess_text(text):  
    # Normalize Persian text  
    text = normalizer.normalize(text)  
  
    for char, replacement in chars_to_mapping.items():  
        text = text.replace(char, replacement)  
  
    # Remove special characters  
    text = re.sub(''.join(map(re.escape, chars_to_ignore)), '', text)  
  
    return text
```

قطعه کد 2: تابع پیش پردازش متن

6. فیلتر کردن

- داده های آموزشی: فیلتر شده تا فقط نمونه های صوتی بین 4 تا 6 ثانیه در مدت زمان داشته باشد

- داده های تست: فیلتر شده تا شامل نمونه های صوتی تا 15 ثانیه در مدت زمان باشد

```
def filter_datasets_by_duration(train_dataset, test_dataset):
    def add_duration(example):
        # Access input_values and sampling_rate directly
        input_values = example['input_values']
        sampling_rate = processor.feature_extractor.sampling_rate # Assuming processor is available in scope
        duration = len(input_values) / sampling_rate
        example['duration'] = duration
        return example

    # Add duration to datasets
    train_dataset = train_dataset.map(add_duration)
    test_dataset = test_dataset.map(add_duration)

    # Filter train dataset (4s to 6s)
    filtered_train = train_dataset.filter(lambda x: 4 <= x['duration'] <= 6)

    # Filter test dataset (0s to 15s)
    filtered_test = test_dataset.filter(lambda x: 0 <= x['duration'] <= 15)

    return filtered_train, filtered_test
```

قطعه کد 3: تابع فیلتر فایل‌ها بر اساس طول زمان

7. واژگان: یک واژگان سفارشی از کاراکترهای منحصر به فرد در مجموعه داده ایجاد شد، از جمله نشانه‌های ویژه برای کاراکترهای ناشناخته "[UNK]" و "[PAD]" padding.

```
def extract_all_chars(batch):
    all_text = " ".join(batch["sentence"])
    vocab = list(set(all_text))
    return {"vocab": [vocab], "all_text": [all_text]}
```

قطعه کد 4: تابع ساخت واژه نامه

8. اندازه

- مجموعه آموزشی: `len(common_voice_train)` نمونه پس از فیلتر

- مجموعه تست: `len(common_voice_test)` نمونه پس از فیلتر

9. ویژگی‌های صوتی

- نرخ نمونه: 16 کیلوهرتز (پس از نمونه برداری مجدد)

- کانال‌ها: مونو

این نسخه مدیریت شده از مجموعه داده فارسی Common Voice پایه ای قوی برای آموزش و ارزیابی مدل wav2vec2 برای تشخیص گفتار فارسی فراهم می کند. مراحل پیش پردازش، سازگاری در قالب صوتی و نمایش متن را تضمین می کند، در حالی که فیلتر کردن به تمرکز آموزش مدل بر روی طول‌های صوتی قابل کنترل کمک می کند. استفاده از یک مجموعه داده در دسترس عموم نیز تکرارپذیری را تضمین می کند و امکان مقایسه با سایر مدل‌های آموزش دیده بر روی همان داده‌ها را فراهم می کند.

هدف ما با تنظیم دقیق مدل wav2vec-large-xlsr-53 بر روی داده‌های فارسی، کمک به پیشرفت فناوری تشخیص گفتار برای این زبان است که به طور بالقوه از طیف گسترده‌ای از برنامه‌ها از دستیار صوتی گرفته تا خدمات رونویسی بهره می‌برد.

نمای کلی معماری مدل

در این تمرین، ما از مدل wav2vec2-large-xlsr-53، یک مدل قدرتمند از پیش آموزش دیده برای وظایف تشخیص گفتار استفاده کردیم. معماری مدل مبتنی بر چارچوب Wav2Vec2 است که توسط فیس بوک AI توسعه یافته است، که به طور خاص برای یادگیری بازنمایی گفتار بین زبانی طراحی شده است.

اجزای اصلی معماری مدل

1. مدل پایه

- ما از مدل از پیش آموزش دیده "facebook/wav2vec2-large-xlsr-53" به عنوان نقطه شروع استفاده کردیم.
- این مدل بر روی 53 زبان آموزش داده شده است که آن را برای سازگاری بین زبانی مناسب می کند.

2. نوع مدل

Wav2Vec2ForCTC: این نوع از مدل Wav2Vec2 برای تشخیص گفتار مبتنی بر طبقه بندی زمانی ارتباطی (CTC) طراحی شده است.

3. استخراج کننده ویژگی

- لایه استخراج ویژگی ورودی صوتی خام را به ویژگی های معنی دار تبدیل می کند.
- در پیاده سازی ما، این لایه (`model.freeze_feature_extractor()`) منجمد شد تا قابلیت های استخراج ویژگی های صوتی از پیش آموزش دیده حفظ شود.

4. Encoder ترانسفورماتور

- هسته مدل متشکل از پشته ای از لایه های ترانسفورماتور است که ویژگی های استخراج شده را پردازش می کند.
- این لایه ها از مکانیسم های خود توجهی (Self-Attention) برای گرفتن وابستگی های دوربرد در سیگنال صوتی استفاده می کنند.

5. لایه خروجی

یک لایه خطی خروجی های ترانسفورماتور را به اندازه واژگان مجموعه داده فارسی ما نگاشت می کند.

6. CTC head

هد CTC (Connectionist Temporal Classification) به مدل اجازه می دهد تا پیش بینی های خروجی را با دنباله صوتی ورودی بدون نیاز به ترازهای صریح تراز کند.

پارامترهای پیکربندی کلیدی

0.1 :attention_dropout -

0.1 :hidden_dropout -

0.0 :feat_proj_dropout -

0.05 :mask_time_prob -

0.1 :layerdrop -

"mean" :ctc_loss_reduction -

اقتباس برای فارسی

- واژگان مدل با استفاده از توکنایزر سفارشی (Wav2Vec2CTCTokenizer) با اندازه واژگان {len(processor.tokenizer)} با زبان فارسی تطبیق داده شد.

- اندازه لایه خروجی مدل از پیش آموزش داده شده برای مطابقت با اندازه واژگان جدید تغییر داده شد.

تکنیک های بهینه سازی

- Gradient Checkpointing: فعال برای کاهش استفاده از حافظه در طول آموزش (model.gradient_checkpointing_enable()).

- آموزش دقیق ترکیبی: از آموزش FP16 برای بهبود کارایی محاسباتی استفاده شده است.

اندازه مدل

- تعداد کل پارامترها: 315499195

- پارامترهای قابل آموزش: 311289019

این معماری با شروع با یک مدل از پیش آموزش داده شده بر روی یک مجموعه داده بزرگ چند زبانه و تنظیم دقیق آن به طور خاص برای تشخیص گفتار فارسی، از قدرت انتقال یادگیری استفاده می کند. استفاده از چارچوب CTC امکان آموزش کارآمد را بدون نیاز به ترازهای صریح و صریح متن صوتی فراهم می کند و آن را برای کارهای تشخیص گفتار سرتاسر مناسب می کند.

روش تنظیم دقیق

رویکرد ما برای تنظیم دقیق مدل wav2vec2-large-xlsr-53 برای تشخیص گفتار فارسی شامل چندین مرحله و استراتژی کلیدی است:

1. آماده سازی داده ها

- مجموعه داده: ما از مجموعه داده فارسی Common Voice 6.1 استفاده کردیم که ترکیب Train و Validation برای آموزش و مجموعه Test برای آزمون است.

- پیش پردازش صدا:

- نمونه برداری مجدد از تمام فایل های صوتی به 16 کیلوهرتز برای مطابقت با نیازهای ورودی مدل.

- داده های آموزشی فیلتر شده برای شامل نمونه هایی بین 4 تا 6 ثانیه در مدت زمان.

- داده های تست فیلتر شده برای شامل نمونه هایی تا 15 ثانیه در مدت زمان.

- پیش پردازش متن:

- عادی سازی متن فارسی با استفاده از کتابخانه hazm.
- نگاشت کاراکتر کاربردی برای استانداردسازی اشکال مختلف کاراکترهای فارسی.
- حذف کاراکترهای خاص و نشانه ها.

2. توکن سازی و استخراج ویژگی

- ایجاد یک واژگان سفارشی از داده های متنی از پیش پردازش شده.
- یک Wav2Vec2CTCTokenizer با این واژگان، از جمله نشانه های ویژه [UNK] و [PAD] پیاده سازی کرد.

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

tokenizer.add_tokens(vocab_list)
```

قطعه کد 5: Tokenization

- استفاده از Wav2Vec2FeatureExtractor برای استخراج ویژگی های صوتی، پیکربندی شده با نرخ نمونه برداری 16000 هرتز.

- ترکیب توکنایزر و استخراج کننده ویژگی در Wav2Vec2Processor برای پردازش ساده داده ها.

3. پیکربندی مدل

- با مدل از پیش آموزش دیده "facebook/wav2vec2-large-xlsr-53"

- لایه خروجی مدل را با اندازه واژگان فارسی سفارشی ما مطابقت می‌دهیم.
- برای حفظ استخراج ویژگی‌های صوتی از پیش آموزش دیده، لایه‌های استخراج ویژگی را فریز می‌کنیم.
- برای بهینه سازی استفاده از حافظه در حین آموزش، کنترل گرادیان را فعال می‌کنیم.

4. راه اندازی آموزش

- بهینه ساز: AdamW با نرخ یادگیری $1e-4$ و کاهش وزن 0.01.
- زمانبندی نرخ یادگیری: برنامه خطی با warmup_steps، با استفاده از warmup_steps 1000.
- اندازه batch: 12 در هر دستگاه، با مراحل انباشتگی گرادیان 2.
- مدت زمان آموزش: 5 epochs.
- سخت افزار: در صورت موجود بودن، از GPU با قابلیت CUDA استفاده می‌شود.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

قطعه کد 6: استفاده از CUDA

5. فرآیند آموزش

- از Hugging Face Trainer API برای آموزش ساده استفاده می‌کنیم.

```

trainer = Trainer(
    model=model,
    data_collator=data_collator,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=common_voice_train,
    eval_dataset=common_voice_test,
    tokenizer=processor.feature_extractor,
    optimizers=(optimizer, lr_scheduler) # Pa
)

```

قطعه کد 7: آموزش

- یک کلاس DataCollatorCTCWithPadding دستی برای دسته بندی و padding کارآمد پیاده سازی کردیم.

```

class DataCollatorCTCWithPadding:

    processor: Wav2Vec2Processor
    padding: Union[bool, str] = True
    max_length: Optional[int] = None
    max_length_labels: Optional[int] = None
    pad_to_multiple_of: Optional[int] = None
    pad_to_multiple_of_labels: Optional[int] = None

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

        batch = self.processor.pad(
            input_features,
            padding=self.padding,
            max_length=self.max_length,
            pad_to_multiple_of=self.pad_to_multiple_of,
            return_tensors="pt",
        )

        # if torch.cuda.is_available():
        #     batch = {k: v.to(device) for k, v in batch.items()}

        with self.processor.as_target_processor():
            labels_batch = self.processor.pad(
                label_features,
                padding=self.padding,
                max_length=self.max_length_labels,
                pad_to_multiple_of=self.pad_to_multiple_of_labels,
                return_tensors="pt",
            )

```

قطعه کد 8: کلاس DataCollatorCTCWithPadding

- استراتژی ارزیابی: ارزیابی هر 150 مرحله انجام می شود.

- ذخیره مدل: پس از هر epoch ذخیره می شود، حداکثر 2 مدل ذخیره شده را نگه می دارد.

6. معیار ارزیابی

- از می‌زان خطای کلمه (WER) به عنوان معیار اولیه برای ارزیابی مدل استفاده شد.

- پیاده سازی تابع `compute_metrics` سفارشی برای محاسبه WER در طول آموزش و ارزیابی.

```
def compute_metrics(pred):  
    pred_ids = pred.predictions.argmax(-1)  
    pred_str = processor.batch_decode(pred_ids)  
    # we do not want to group tokens when computing the metrics  
    label_ids = pred.label_ids  
    # replace padding with -100 to compute the correct WER  
    label_ids[label_ids == -100] = processor.tokenizer.pad_token_id  
    label_str = processor.batch_decode(label_ids, group_tokens=False)  
    wer = wer_metric.compute(predictions=pred_str, references=label_str)  
    return {"wer": wer}
```

قطعه کد 9: محاسبه WER

7. تکنیک‌های تنظیم دقیق

- آموزش دقیق ترکیبی: آموزش FP16 را برای بهبود کارایی محاسباتی فعال شد.

- Gradient Accuulation: برای افزایش موثر اندازه دسته بدون افزایش مصرف حافظه استفاده می شود.

- انجماد لایه: لایه‌های استخراج ویژگی را در حالی که بقیه مدل را دقیق تنظیم می کنید، منجمد نگه می دارد.

8. پس از آموزش

- مدل و پردازنده تنظیم شده نهایی را برای استفاده و استقرار در آینده ذخیره می‌کنیم.

این روش از یادگیری انتقالی با شروع با یک مدل چندزبانه قدرتمند و تطبیق آن به طور خاص برای فارسی استفاده می‌کند. پیش پردازش دقیق مجموعه داده فارسی، همراه با استراتژی‌های تنظیم دقیق هدفمند، با هدف ایجاد یک مدل تشخیص گفتار با کارایی بالا متناسب با ویژگی‌های منحصر به فرد زبان فارسی است.

نتایج و تحلیل

- جملات اولیه نمونه

sentence	
0	پنجاه و هفت، پنجاه و هشت، پنجاه و نه
1	او اینجا در تعطیلات است
2	چطور میتونم راحت باشم؟
3	از این گوش میگیره از اون گوش در میکنه
4	برنشست
5	برای دیگران توضیح دهد
6	مصر
7	این قطار مستقیم است؟
8	بیماری کم خونی داسی شکل
9	هیچ دستمالی دارید؟

نتایج 1: جملات بیناست

این جملات شامل " "، علامت‌های نگارشی و ... هستند و هنوز token بندی و نرمال نشده اند.

- ساخت واژه نامه

```
{'0': 'ع',
' ': 1,
'2': 'آ',
'T': 3,
'&': 4,
'5': 'غ',
'6': 'ی',
'7': 'ه',
'F': 8,
'9': 'ژ',
'H': 10,
'G': 11,
'12': 'ز',
'13': 'ث',
'K': 14,
'A': 15,
'E': 16,
'M': 17,
'S': 18,
'": 19,
'I': 20,
'21': 'ا',
'22': 'ل',
'D': 23,
'24': 'ش',
...
'49': 'و',
'50': 'ن',
'51': 'گ',
'U': 52,
'53': 'ک'}
```

نتایج 2: واژه نامه ساخته شده

می‌بینیم که لغت نامه به طور صحیح ساخته نشده. چرا که طول آن می‌بایست 35 باشد. اما الان 53 (با سه توکن unk و pad و | می‌شود 56) است. این مشکل می‌تواند در ادامه در یادگیری مدل ما تاثیر بگذارد.

- تغییر فرکانس نمونه برداری

```
common_voice_train = common_voice_train.cast_column("audio", Audio(sampling_rate=16_000))
common_voice_test = common_voice_test.cast_column("audio", Audio(sampling_rate=16_000))

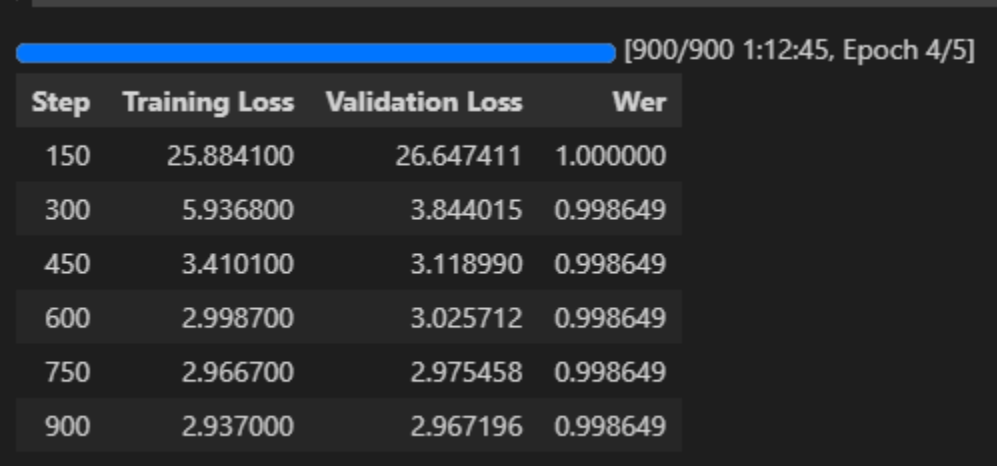
common_voice_train[0]["audio"]

{'path': '/root/.cache/huggingface/datasets/downloads/extracted/c5031a95ac0e40545f16661cce5727f9d20ab85ee4a9fa4',
 'array': array([ 9.09494702e-13, -1.81898940e-12,  1.00044417e-11, ...,
                  5.28323289e-06,  4.76915011e-06, -1.47444371e-06]),
 'sampling_rate': 16000}
```

نتایج 3: تعبیر فرکانس صوت ها

از 48 کیلو هرتز به 16 کیلو هرتز.

- آموزش



A blue progress bar is shown at the top of the table, indicating the training progress. The text '[900/900 1:12:45, Epoch 4/5]' is displayed to the right of the bar.

Step	Training Loss	Validation Loss	Wer
150	25.884100	26.647411	1.000000
300	5.936800	3.844015	0.998649
450	3.410100	3.118990	0.998649
600	2.998700	3.025712	0.998649
750	2.966700	2.975458	0.998649
900	2.937000	2.967196	0.998649

نتایج 4: آموزش مدل

- تست

[652/652 05:57]

```
{'eval_loss': 2.967195510864258,  
'eval_wer': 0.9986490205398915,  
'eval_runtime': 360.7933,  
'eval_samples_per_second': 14.446,  
'eval_steps_per_second': 1.807,  
'epoch': 4.986149584487535}
```

نتایج 5: تست مدل

مشاهده می‌شود که مقدار loss در طول آموزش بسیار کم شده. اما مقدار WER به عنوان واحد ارزیابی مدل، تغییر چنانی نکرده و مقدار بالایی دارد.

این مشکل می‌تواند به دو دلیل باشد:

- 1- فیلتر داده‌های آموزشی بر اساس زمان که تعداد آن‌ها را بسیار کمتر کرده (تقریباً یک سوم)
- 2- توکن بندی ناصحیح و ساخت لغت نامه اشتباه که می‌تواند دلیل اصلی این موضوع باشد.