دانشگاه صنعتي اميركبير
(پلي تكنيك تهران)

دانشكده مهندسي كامپيوتر و
فناوري اطلاعات

به نام خدا

تمرین دوم فهم زبان

زینب خالوندی

شماره دانشجویی:

99131007

آذر 1400

مقدمه

در این تمرین با دو مسئله‌ی تشخیص قصد و پرکردن شکاف مواجه هستیم. تشخیص قصد دسته‌بندی در سطح جمله و پرکردن شکاف دسته‌بندی در سطح کلمه است. با این که این دو مسئله باید هرکدام بصورت جداگانه حل شوند اما در حل هر کدام از آن‌ها می‌توان از اطلاعات مربوط دیگری استفاده کرد. زیرا تقریبا می‌توان گفت که برای هر قصد شکاف‌های منحصر به آن تعریف می‌شود و بین شکاف‌های قصدهای مختلف هم‌پوشانی کمی وجود دارد.

برای حل این دو مسئله از دو روش bilstm با آموزش غیر همزمان و شبکه برت استفاده شده‌است.

داده‌ها

داده‌های بصورت سه‌تایی‌هایی مرتب از کلمات جمله، برچسب هر کلمه برای تشخیص شکاف و برچسب کل جمله برای تشخیص قصد استخراج شده‌اند.

در کل داده‌ها در از نظر دسته‌بندی قصد در دوازده کلاس توزیع شده‌اند. این کلاس‌ها به شرح زیر هستند.

```
intents_dict = {'weather/find': 0,
                'alarm/set_alarm': 1,
                'alarm/show_alarms': 2,
                'reminder/set_reminder': 3,
                'alarm/modify_alarm': 4,
                'weather/checkSunrise': 5,
                'weather/checkSunset': 6,
                'alarm/snooze_alarm': 7,
                'alarm/cancel_alarm': 8,
                'reminder/show_reminders': 9,
                'reminder/cancel_reminder': 10,
                'alarm/time_left_on_alarm': 11}
```

شکاف‌ها دارای 28 برچسب مختلف هستند. این برچسب‌ها به شرح زیر هستند.

```
slots_dict = {'NoLabel': 0,
 'B-weather/noun': 1,
 'I-weather/noun': 2,
 'B-location': 3,
 'I-location': 4,
 'B-datetime': 5,
 'I-datetime': 6,
 'B-weather/attribute': 7,
 'I-weather/attribute': 8,
 'B-reminder/todo': 9,
```

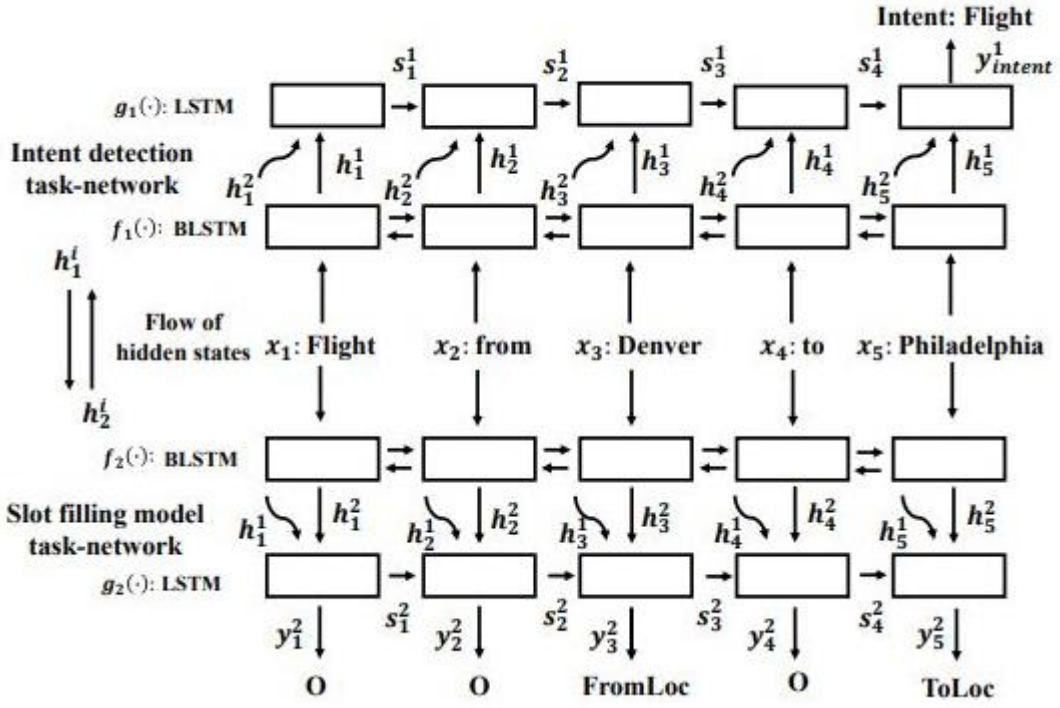
```
'I-reminder/todo': 10,
'B-alarm/alarm_modifier': 11,
'B-reminder/noun': 12,
'B-reminder/recurring_period': 13,
'I-reminder/recurring_period': 14,
'B-reminder/reference': 15,
'I-reminder/noun': 16,
'B-reminder/reminder_modifier': 17,
'B-timer/noun': 18,
'I-reminder/reference': 19,
'B-negation': 20,
'B-timer/attributes': 21,
'B-news/type': 22,
'I-reminder/reminder_modifier': 23,
'B-weather/temperatureUnit': 24,
'I-alarm/alarm_modifier': 25,
"B-demonstrative_reference":26,
"I-demonstrative_reference":27,
"PAD":28}
```

نحوه‌ی جداسازی کلمات به همان ترتیبی است که برچسب‌های شکاف به آن‌ها منتسب شده‌است. کلماتی که عدد هستند به کلمه‌ی digit نگاشت شده‌اند.

روش اول: bilstm آموزش غیرهمزمان[1]

در این حالت از دو شبکه‌ی مبتنی بر bilstm استفاده شده است. یکی برای تشخیص قصد و یکی برای پر کردن شکاف. معماری شبکه به شکل زیر است.

[1] https://github.com/ray075hl/Bi-Model-Intent-And-Slot



داده‌ها:

در این بخش همه‌ی ورودی‌ها به یک طول درآمده‌اند. با توجه به مقدار بیشینه، میانگین و انحراف از معیار طول تمام دادهای آموزش طول 15 برای دنباله‌های ورودی انتخاب شده‌است و بخش اضافی دنباله‌های طولانی‌تر نادیده گرفته شده است و به دنباله‌های کوتاه‌تر یک جز ثابت(PAD) اضافه شده است.

آموزش شبکه:

در این روش ابتدا ورودی به اینکودر هر دو شبکه وارد می‌شود و بردارهای نهان به ازای این ورودی محاسبه می‌شوند. سپس از این بردارهای نهان هر دو شبکه برای یک گام آموزش و بروز رسانی وزن‌های بخش پرکردن شکاف استفاده می‌شود. مقدار تابع هزینه‌ی به ازای مسئله‌ی پرکردن شکاف محاسبه می‌شود و وزن‌ها بروز می-شوند. تابع هزینه تابع کراس آنتروپی است که به ازای خروجی تمام کلمات جمله محاسبه می‌شود.

پس از یک گام آموزش بخش پرکردن شکاف، بخش تشخیص قصد آموزش می‌بیند و آموزش کل سیستم یک مرحله کامل می‌شود.

آزمایشات:

آزمایش یک:

```
max_len = 15
learning_rate = 0.001
total_epoch = 80
batch = 128
DROPOUT = 0.1
embedding_size = 100
lstm_hidden_size = 80
```

```
Epoch: 80
Intent Val Acc: 63.5820
Slot F1 score: 95.3600
********************
Best Intent Acc: 64.4189 at Epoch: [56]
Best F1 score: 0.9569 at Epoch: [43]
```

آزمایش دوم:

max_len = 15

learning_rate = 0.001

total_epoch = 60

batch = 256

DROPOUT = 0.1

embedding_size = 200

lstm_hidden_size = 100

Epoch: 60
Intent Val Acc: 63.3668
Slot F1 score: 95.5752
****
Best Intent Acc: 64.4429 at Epoch: [47]

Best F1 score: 0.9581 at Epoch: [23]

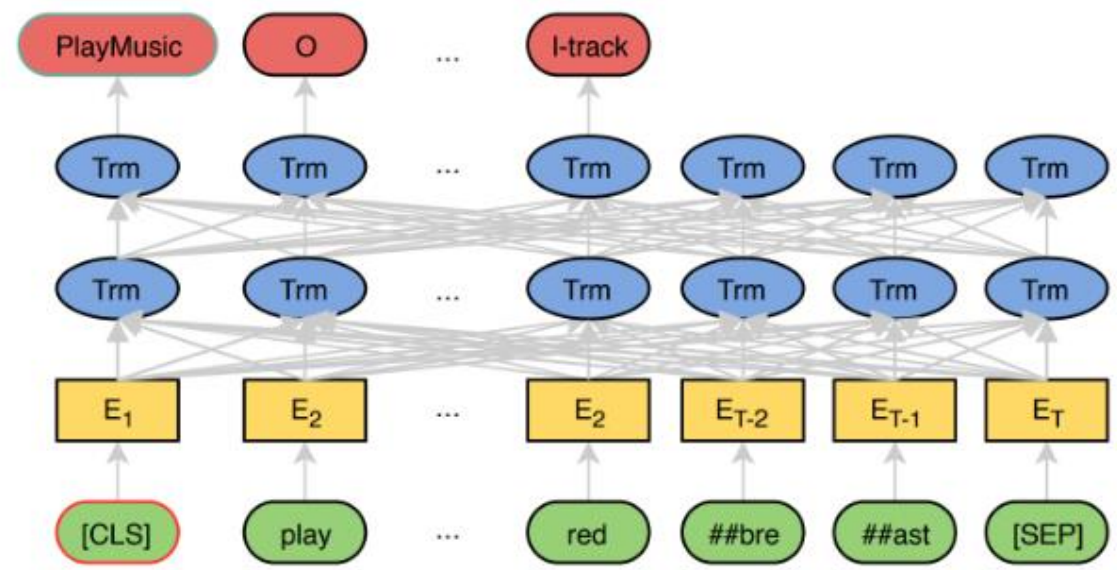
نتیجه‌ی شبکه بر روی داده‌های آزمایش

F1 score slot detection: 0.9560375826470245

Best Intent Acc: 0.6425423335652981

روش دوم: JoinBert

در این بخش از برت که بخش اینکودر یک شبکه‌ی ترنسفرمری استفاده شده‌است. بدین منظور از مدل‌های از پیش آموزش داده شده huggingface استفاده شده است. دسته‌بندی تشخیص قصد بر روی نشانه‌ی cls انجام شده است و برای پرکردن شکاف از دسته‌بندی برروی خروجی هر توکن انجام شده است.



داده‌ها:

برای تجزیه‌ی کلمات به زیرکلمه‌ها، از tokenizer های از پیش آموزش داده شده استفاده شده است. بر اساس مدل پیش آموزش داده شده‌ایی که استفاده شده است ار tokenizer آن استفاده شده است. برای تبدیل جملات ورودی به بردارهای عددی مطابق با id توکنایزر هر کلمه به صورت جداگانه به به زیر کلمات تبدیل شده است. برای هماهنگی زیرکلمات با برچسب های کلمات، برچسب هر کلمه به اولین زیرکلمه‌ی آن نسبت داده شده است و برچسب باقی کلمات برچسب pad در نظر گرفته شده است. به ابتدای جملات cls افزوده شده است.

شبکه:

شبکه‌ی بکاربرده شده برای این مسئله به شکل زیر است. یک مدل برت از پیش آموزش داده شده استفاده شده است. خروجی لایه‌ی آخر این شبکه برای هر کلمه استخراج شده است. بر روی خروجی ابتدایی "cls" یک لایه‌ی تمام متصل برای دسته‌بندی قصد اعمال شده است. و بر روی هر خروجی دیگر یک لایه تمام متصل برای پر کردن هر شکاف اعمال شده است.

تابع هزینه‌ی این دو دسته‌بندی از نوع کراس آنتروپی است و برای دو وظیفه‌ی تشخیص قصد و پرکردن شکاف مجموع هزینه محاسبه شده است.

آزمایشات:

برای یافتن تنظیمات بهتر آزمایشات زیر انجام شده است.

طول دنباله‌ی ورودی با توجه به بیشترین طول دنباله‌ها، کمترین طول و دامنه‌ی تغییرات آن‌ها انتخاب شده است.

آزمایش یک:

max_seq_len = 15

learning_rate = 5e-5

num_train_epochs = 4

train_batch_size = 32

eval_batch_size = 64

dropout_rate = 0.1

دقت ارزیابی

Intent Accuracy : 98.7

Fscore slot detection: 97.1

آزمایش دو:

```
max_seq_len = 15
learning_rate = 5e-6
num_train_epochs = 4
train_batch_size = 64
eval_batch_size = 256
dropout_rate = 0.3
```

دقت ارزیابی

```
accuracy score intent detection 0.9
slot fscore 0.9411764705882353
```

با افزایش مقدار dropout و کاهش نرخ یادگیری دقت کاهش یافته است.

در نهایت شبکه با تنظیمات آزمایش یک آموزش داده شده و دقت تست به شکل زیر بدست آمده است.

accuracy score intent detection 0.9939682171441828
slot fscore 0.9783520032987424

طراحی آزمایشی با هدف افزایش دقت پرکردن شکاف:

به منظور اینکه هر کلمه بصورت مستقیم از اطلاعات بدست آمده از کل جمله و قصد جمله استفاده کند، خروجی توکن cls به خروجی هر کلمه اضافه شده است و به عنوان ورودی به دسته‌بند هر کلمه داده می‌شود.

شبکه با این تغییر آموزش داده شده است اما در نتایج تغییر محسوسی ایجاد نشده است. دلیل این عدم بهبود احتملا بخاطر مکانیزم توجه است که برای هر کلمه اطلاعات تمام جمله را در نظر میگیرد و به اندازه‌ی کافی به تمام بافت جمله توجه می‌شود.

نتایج ارزیابی و تست این آزمایش به نحوه زیر است:

آزمایش دو:

تنظیمات مطابق آزمایش یک

ارزیابی

accuracy score intent detection 0.9865
slot fscore 0.9710836608646827

تست

accuracy score intent detection 0.9937362254958821
slot fscore 0.9784879725085911

مقایسه و نتیجه‌گیری

طبق نتایج بدست آمده نتیجه‌ی روش اول و استفاده از lstm و آموزش غیرهمزمان نسبت به استفاده از برت نتیجه‌ی خوبی تولید نمی‌کند.

عملکرد خوب مدل برت با توجه به اینکه مبتنی بر مکانیزم توجه و استفاده از مدل از پیش آموزش داده شده است و با داده‌های آموزش fine tune می‌شود، قابل انتظار است، به این دلیل که مدل به خوبی ساختار جملات را از نظر دستوری، معنایی و نحوه‌ی قرارگیری در کنارهم آموزش می‌بیند. در مورد تفاوت اندکی که در دقت و امتیاز تشخیص قصد و پرکردن شکاف است به دلیل این است که تعداد کلاس‌های قصد کمتر است و تنوع کلاس‌ها در شکاف‌ها بیشتر است و تعداد کلماتی که برچسب nolabel دارند زیاد است.

درمورد مدل lstm یکی از دلایل کم بودن دقت در تشخیص قصد را می‌توان به این دلیل دانست که وظیفه‌ی تشخیص قصد از اطلاعات بدست آمده از گام قبلی آموزش پرکردن شکاف استفاده می‌کند و این بردارهای تولید شده ممکن است بخصوص در گام‌های اولیه خطا داشته باشند و مدل تشخیص قصد را به اشتباه بیندازد. همین موجب کم شدن دقت شود.

در مورد کم بودن امتیاز پرکردن شکاف در مقایسه با مدل برت می‌توان گفت که lstm نمی تواند به اندازه ی برت بافت جمله را یاد بگیرد، به دلیل اینکه آموزش اولیه را ندارد و در واقع باید از پایه آموزش ببیند.

از نظر زمانی مدل برت نسبت به lstm کندتر است که دلیل آن پارامترهای بسیار بیشتر آن است، اما در عوض در تعداد تکرار کمتری به دقت مورد انتظار می‌رسد.