تکلیف سری ششم درس پردازش زبان‌های طبیعی
فاطمه نادی ۸۱۰۱۰۱۲۸۵

بخش اول – ربات پاسخ‌گو به پرسش‌های پرتکرار 2
بخش ۱: آماده‌سازی داده‌ها..... 2
Sub-intent..... 2
Multi-intent 7
بخش ۲ : انجام تنظیمات و آموزش ربات 10
Sub-intent..... 10
Multi-intent..... 11
سوال ۳: ارزیابی ربات 13
Sub-intent..... 13
Sub-intent - ParsBert with 50 epochs 13
Sub-intent - LabSE with 50 epochs..... 15
Sub-intent - LabSE with 100 epochs..... 17
Sub-intent - LabSE with 200 epochs..... 19
Multi-intent 21
Multi-intent - ParsBert with 50 epochs 21
Multi-intent - LabSE with 50 epochs 23
Multi-intent - LabSE with 100 epochs 25
Multi-intent - LabSE with 200 epochs 27
مقایسه نهایی 28
بخش دوم – استخراج مقادیر ارزش‌ها 29

بخش اول – ربات پاسخ‌گو به پرسش‌های پرتکرار

بخش ۱: آماده‌سازی داده‌ها

دو رویکرد مختلف برای آماده‌سازی دیتا مطرح شده است که به تفضیل آماده‌سازی هر کدام را شرح خواهیم داد:

Sub-intent

رویکرد اول قرار دادن همه intentها زیر مجموعه یک کلاس بزرگ‌تر به نام faq که هر پرسش‌وپاسخ زیر مجموعه کلاس faq قرار خواهد گرفت.
هر زیرکلاس با id متفاوت و به وسیله / و نام کلاس که در فایل‌ها مختلف به همین نام ثبت شده است که در ادامه نحوه آماده‌سازی هر فایل شرح داده خواهد شد:

1. NLU

```python
def generate_nlu_sub_intent(dataframe, file_path, operation_mode):
    with open(file_path, "w", encoding="utf-8") as output_file:
        output_file.write('version: "3.1"\n\n')
        output_file.write("nlu:\n")
        for index, record in dataframe.iterrows():
            output_file.write(f"\n- intent: faq/Q{record[0]}\n")
            output_file.write("  examples: |\n")
            if operation_mode == "train":
                for i in range(15):
                    output_file.write(f'    - "{record[7+i]}"\n')
            else:
                for i in range(5):
                    output_file.write(f'    - "{record[7+i]}"\n')
```

فایل `nlu.yml` در Rasa، یکی از فایل‌های اصلی و مهم برای توسعه مدل پردازش زبان طبیعی (Natural Language Understanding - NLU) است. این فایل برای تعریف داده‌های آموزشی مربوط به فهم و تحلیل دستورات کاربر استفاده می‌شود. در واقع، فایل `nlu.yml` شامل مجموعه‌ای از نمونه‌های داده است که شامل متن دستورهای کاربر، نام اکشن‌ها و همچنین نام entityها (موجودیت‌ها) است.

در Rasa، فهم دستورات کاربر به دو بخش اصلی تقسیم می‌شود: فهم نوع دستور (Intent Classification) و استخراج اطلاعات مفید (Entity Extraction). فایل `nlu.yml` برای تعریف این دو مورد استفاده می‌شود.

در بخش فهم نوع دستور، نمونه‌هایی از دستورات مختلف که کاربران می‌توانند وارد کنند، تعریف می‌شود. برای هر نمونه، یک نام (Intent) تعیین می‌شود که نوع دستور را مشخص می‌کند که در واقع سوالاتی است که سمت کاربر مطرح می‌شود، به عنوان مثال، یک نمونه از یک دستور ممکن در فایل `nlu.yml` می‌تواند به صورت زیر باشد:

```
- intent: greet
  examples: |
    - Hello
    - Hi
    - Hey there
```

در این مثال، نام دستور (intent) در اینجا greet است و نمونه‌هایی از دستورات ممکن برای این نوع دستور آورده شده است.
هر دستور شامل تعدادی مثال بوده که باتوجه به آن مدل آموزش می‌بیند، این بخش بدین منظور اعمال می‌شود که اگر مدل سوالاتی شبیه به آنچه که در این‌جا مطرح شده است را دید بتواند پاسخ متناظر با آن‌چه در ادامه در فایل domain است را به عنوان خروجی برگرداند.

در بخش استخراج اطلاعات مفید، موجودیت‌هایی که در دستور کاربر ممکن است مورد نیاز باشند، را تعریف می‌شود. به عنوان مثال، اگر در دستور کاربر نیاز به استخراج نام و نام خانوادگی باشد، یک موجودیت (Entity) به نام "person" با تگ‌های متناظر با آن تعریف خواهد شد.

فرمت نوشتار فایل test, train کاملا یکسان بوده و بدین صورت ذخیره می‌شود که ابتدا ورژن و سپس keyها درج می‌شود و در ادامه برای هر key تعدادی زیربخش که شامل سوالات مربوطه است هر کدام در سطری جداگانه نوشته خواهد شد.

```yaml
version: "3.1"

nlu:

- intent: faq/Q1
  examples: |
    - "با استفاده از کد دستوری چیکار کنم؟ برای فعال سازی سرویس gprs"
    - "چه شماره ای باید بگیرم؟ برای فعالسازی سرویس gprs"
    - "چه کدی رو باید بزنم؟ برای فعالسازی سرویس gprs"
    - "با استفاده از کد دستوری رو فعال کنم؟ چجوری سرویس اینترنت gprs"
    - "باید بگیرم؟ چه شماره ای برای فعالسازی سرویس اینترنت gprs"
    - "باید ارسال کنم؟ چه کدی برای فعالسازی سرویس اینترنت gprs"
    - "رو با استفاده از کد دستوری چجوری فعال کنم؟ gprs"
    - "سرویس اینترنت فعال سازیش با استفاده از کد دستوری چجوریه؟"
    - "کد فعالسازی سرویس اینترنت چیه؟"
    - "چیکار باید بکنم تا سرویس اینترنتم با استفاده از کد دستوری فعال بشه؟"
    - "فعال بشه با استفاده از کد دستوری باید چیکار کنم؟ میخوام سرویس اینترنت gprs"
    - "سرویس اینترنتم چی جوری با استفاده از کد دستوری فعال میشه؟"
    - "واسه فعال شدن سرویس اینترنتم با کد دستوری چیکار باید بکنم؟"
    - "برای فعال شدن سرویس اینترنت چه کدی رو باید بگیرم؟"
    - "با استفاده از کد دستوری کاری باید بکنم؟ برای فعال شدن اینترنت gprs"

- intent: faq/Q2
  examples: |
    - "از طریق پیامک چیکار کنم؟ برای فعالسازی سرویس gprs"
    - "به چه شماره ای باید پیامک بدم؟ برای فعال سازی سرویس gprs"
    - "به چه شماره ای باید اس ام اس بزنم؟؟ برای فعالسازی سرویس gprs"
    - "از طریق پیامک اس ام اس رو فعال کنم؟ چجوری سرویس اینترنت gprs"
    - "باید پیامک بدم؟ به کجا برای فعال سازی سرویس اینترنت gprs"
    - "ارسال کنم؟ باید sms به چه شماره ای برای فعالسازی سرویس اینترنت gprs"
    - "رو از طریق پیامک چجوری فعال کنم؟ gprs"
    - "چجوریه؟ سرویس اینترنت فعال سازیش از طریق پیامک Sms"
    - "فعال سازی سرویس اینترنت چیه؟ sms"
    - "بدم تا سرویس اینترنتم فعال بشه؟ به چه شماره ای پیامک اس ام اس sms"
    - "فعال بشه از طریق پیامک باید چیکار کنم؟ میخوام سرویس اینترنت gprs"
    - "سرویس اینترنتم از طریق پیامک چیجوری فعال میشه؟"
    - "واسه فعال شدن سرویس اینترنتم از طریق پیامک اس ام اس چیکار باید بکنم؟"
    - "برای فعال شدن سرویس اینترنت چه اس ام اسی باید بدم؟"
    - "از طریق پیامک کاری باید بکنم؟ برای فعال شدن اینترنت gprs"
```

2. Domain

```
def generate_domain_sub_intent(dataframe, file_path):
    with open(file_path, "w", encoding="utf-8") as output_file:
        output_file.write('version: "3.1"\n\n')
        output_file.write("intents:\n")
        output_file.write("    - faq\n\n")
        output_file.write("responses:\n")
        for index, record in dataframe.iterrows():
            output_file.write(f"    utter_faq/Q{record[0]}:\n")
            output_file.write(f'    - text: "{record[1]}"\n')
        output_file.write("\nactions:\n")
        output_file.write("    - utter_faq\n")
```

فایل domain در Rasa برای تعریف و مدیریت دامنه (domain) یک سیستم گفتگویی استفاده می‌شود. دامنه شامل اطلاعاتی است که در مورد تعاملات و قابلیت‌های سیستم گفتگویی است. این فایل حاوی اطلاعاتی مانند اسامی اکشن‌ها، انواع تعاملات کاربر، انواع entityها، انواع اعمال‌کننده‌ها (form actions)، چالش‌ها (utterances) و پاسخ‌ها (responses) است.
پاسخ‌های مناسب برای هر کلاس در این فایل مقداردهی می‌شود بدین صورت که پاسخ مناسب هر کلاس تعریف شده در فایل NLU یک کلاس با همان اسم به اضافه uter به ابتدای نام آن تعریف می‌شود. پس از مشخص کردن intentهایی که قرار است در این فایل به آن‌ها پاسخ داده‌شود برای آموزش مدل مورد استفاده قرار می‌گیرد.

برای آماده‌سازی دیتاست و ساخت فایل domain در Rasa، می‌توان مراحل زیر را دنبال کرد:
تعریف چالش‌ها (utterances) و پاسخ‌ها (responses) بدین صورت است که باید پیام‌های چالش و پاسخ‌های مرتبط را برای سیستم گفتگویی تعریف شود. چالش‌ها و پاسخ‌ها را در بخش "responses" فایل domain تعریف می‌شو که با اضافه کردن uter_ مورد استفاده قرار می‌گیرد.

3. Stories

```
def generate_stories_sub_intent(dataframe, file_path):
    with open(file_path, "w", encoding="utf-8") as output_file:
        output_file.write('version: "3.1"\n\n')
```

فایل Stories در فریمورک Rasa برای تعریف رفتارهای مورد انتظار یک Chat Bot استفاده می‌شود. این فایل‌ها شامل توالی اقدامات (اعمال کاربری و پاسخ ربات) در یک دستورالعمل چت هستند و نحوه تعامل بین کاربر و ربات را مشخص می‌کنند.

در فایل Stories، هر خط یک داستان را نشان می‌دهد. هر داستان می‌تواند شامل یک یا چند فعالیت کاربر و یک پاسخ ربات باشد. در واقع، داستان‌ها مشخص می‌کنند که ربات در واکنش به ورودی کاربر چه اقداماتی را انجام می‌دهد.

یک داستان در Rasa معمولاً با عبارتی شروع می‌شود که مشخص می‌کند کاربر چه فعالیتی انجام می‌دهد. سپس در خطوط بعدی، واکنش ربات به فعالیت کاربر تعریف می‌شود. برای هر مرحله از تعامل، شرایط قبلی (مثلاً چه وضعیتی در حال حاضر وجود داشته باشد) و اقدامات مورد انتظار را مشخص کنید.

در زیر یک مثال از یک فایل Stories در Rasa آورده شده است:

```
## story_example
* greet
  - utter_greet
* ask_restaurant
  - utter_ask_cuisine
* inform{"cuisine": "Italian"}
  - action_search_restaurant
  - utter_display_results
* affirm
  - utter_book_table
```

در این مثال، داستان شروع می‌شود با عملیات کاربر "greet" (سلام کردن) و سپس ربات با "utter_greet" پاسخ می‌دهد. سپس کاربر پرسشی در مورد رستوران می‌پرسد ("ask_restaurant") و ربات با "utter_ask_cuisine" پاسخ می‌دهد و کاربر اطلاعات رستوران مورد نظر خود را اعلام می‌کند ("inform") و ربات با اقداماتی مانند "action_search_restaurant" و "utter_display_results" واکنش نشان می‌دهد. در نهایت، کاربر با "

affirm" پاسخ می‌دهد و ربات با "utter_book_table" پاسخ می‌دهد.

با نوشتن چنین داستان‌هایی، شما می‌توانید تعامل کاربر و ربات را تعریف کنید و مدل Rasa می‌تواند با استفاده از این داستان‌ها و آموزش بر روی آن‌ها، مکانیزمی برای پیش‌بینی رفتار ربات در موقعیت‌های مشابه ایجاد کند و کمک می‌کند تا مسیر برای دادگان دیده نشده در مدل یادگیری شود.

4. Rules

```
def generate_rules_sub_intent(dataframe, file_path):
    with open(file_path, "w", encoding="utf-8") as output_file:
        output_file.write('version: "3.1"\n\n')
```

در Rasa، فایل rules برای تعریف قوانین مشخصی استفاده می‌شود که می‌تواند ترکیبی از محدودیت‌ها و اقدامات باشد. برای آماده‌سازی دیتاست برای فایل rules در Rasa، شما نیاز دارید قوانین مورد نظر خود را تعریف کنید. قوانین شامل دو بخش اصلی هستند: شرط‌ها (conditions) و اقدامات (actions).

- شرط‌ها معمولاً بر اساس ویژگی‌های فعلی یا پیشینه‌ای از گفتگو تعریف می‌شوند. برای تعریف شرط‌ها، از گزینه‌هایی مانند slot value، form action یا custom actions استفاده کرد.
- اقدامات نشان می‌دهند که چه کاری باید انجام شود در صورتی که شرط‌ها برقرار باشند. برای تعریف اقدامات، از send message، run action یا set slot استفاده می‌شود.

Multi-intent

رویکرد دوم هر سوال یک intent جداگانه بوده و response هر یک نیز جداگانه تعریف می شود.

1. NLU

```python
def generate_nlu_multi_intent(dataframe, file_path, mode):
    with open(file_path, "w", encoding="utf-8") as file:
        file.write('version: "3.1"\n\n')
        file.write("nlu:\n")
        for index, record in dataframe.iterrows():
            file.write(f"\n- intent: net{record[0]}\n")
            file.write("  examples: |\n")
            if mode == "train":
                for i in range(15):
                    file.write(f'    - "{record[7+i]}"\n')
            else:
                for i in range(5):
                    file.write(f'    - "{record[7+i]}"\n')
```

فایل Natural Language Understanding (NLU) در چارچوب Rasa برای یادگیری و تفسیر دستورات کاربر از طریق پردازش زبان طبیعی استفاده می‌شود. در مواردی که ما به دنبال شناسایی چندین نوع از دستورات (multi-intent) هستیم می‌توان از قابلیت Multi-Intent Detection در Rasa استفاده کرد.

برای نوشتن فایل NLU برای multi-intent، باید برچسب‌های مربوط به هر intent را در یک فایل باشد. برای این کار، می‌توان از قالب Markdown یا قالب YAML استفاده کرد. در ادامه، نمونه‌ای از نحوه نوشتن فایل NLU برای multi-intent در قالب YAML به شرح زیر است:

```
nlu:
- intent: greet
  examples: |
    - Hello
    - Hi
    - Hey

- intent: goodbye
  examples: |
    - Goodbye
    - Bye
    - See you later
```

در این مثال، چندین intent مختلف را تعریف شده، مانند `goodbye` ،`greet`، و برای هر یک نمونه‌هایی از جملاتی که ممکن است کاربر بگوید، آورده شده است.
و در نهایت فرم نوشته شده این فایل به شکل زیر خواهد بود:

```
version: "3.1"

nlu:

- intent: Q1
  examples: |
    - "با استفاده از کد دستوری چیکار کنم؟ gprs برای فعال سازی سرویس"
    - "چه شماره ای باید بگیرم؟ gprs برای فعالسازی سرویس"
    - "چه کدی رو باید بزنم؟ gprs برای فعالسازی سرویس"
    - "با استفاده از کد دستوری رو فعال کنم؟ gprs چجوری سرویس اینترنت"
    - "باید بگیرم؟ gprs چه شماره ای برای فعالسازی سرویس اینترنت"
    - "باید ارسال کنم؟ gprs چه کدی برای فعالسازی سرویس اینترنت"
    - "رو با استفاده از کد دستوری چجوری فعال کنم؟ gprs"
    - "سرویس اینترنت فعال سازیش با استفاده از کد دستوری چجوریه؟"
    - "کد فعالسازی سرویس اینترنت چیه؟"
    - "چیکار باید بکنم تا سرویس اینترنتم با استفاده از کد دستوری فعال بشه؟"
    - "فعال بشه با استفاده از کد دستوری باید چیکار کنم؟ gprs میخوام سرویس اینترنت"
    - "سرویس اینترنتم چی جوری با استفاده از کد دستوری فعال میشه؟"
    - "واسه فعال شدن سرویس اینترنتم با کد دستوری چیکار باید بکنم؟"
    - "برای فعال شدن سرویس اینترنت چه کدی رو باید بگیرم؟"
    - "با استفاده از کد دستوری کاری باید بکنم؟ gprs برای فعال شدن اینترنت"

- intent: Q2
  examples: |
    - "از طریق پیامک چیکار کنم؟ gprs برای فعالسازی سرویس"
    - "به چه شماره ای باید پیامک بدم؟ gprs برای فعال سازی سرویس"
    - "به چه شماره ای باید اس ام اس بزنم؟؟ gprs برای فعالسازی سرویس"
    - "از طریق پیامک اس ام اس رو فعال کنم؟ gprs چجوری سرویس اینترنت"
    - "باید پیامک بدم؟ gprs به کجا برای فعال سازی سرویس اینترنت"
    - "ارسال کنم؟ sms باید gprs به چه شماره ای برای فعالسازی سرویس اینترنت"
    - "رو از طریق پیامک چجوری فعال کنم؟ gprs"
    - "چجوریه؟ Sms سرویس اینترنت فعال سازیش از طریق پیامک"
    - "فعال سازی سرویس اینترنت چیه؟ sms"
    - "بدم تا سرویس اینترنتم فعال بشه؟ sms به چه شماره ای پیامک اس ام اس"
    - "فعال بشه از طریق پیامک باید چیکار کنم؟ gprs میخوام سرویس اینترنت"
    - "سرویس اینترنتم از طریق پیامک چیجوری فعال میشه؟"
    - "واسه فعال شدن سرویس اینترنتم از طریق پیامک اس ام اس چیکار باید بکنم؟"
    - "برای فعال شدن سرویس اینترنت چه اس ام اسی باید بدم؟"
    - "از طریق پیامک کاری باید بکنم؟ gprs برای فعال شدن اینترنت"
```

2. Domain

```python
def generate_domain_multi_intent(dataframe, file_path):
    with open(file_path, "w", encoding="utf-8") as file:
        file.write('version: "3.1"\n\n')
        file.write("intents:\n")

        for index, record in dataframe.iterrows():
            file.write(f"    - q{record[0]}\n")

        file.write("\nresponses:\n")
        for index, record in dataframe.iterrows():
            file.write(f"    utter_Q{record[0]}:\n")
            file.write(f'    - text: "{record[1]}"\n')
```

همانند قسمت قبل در این فایل نیز پاسخ مربوطه به سوالاتی که در فایل NLU مطرح شده قرار می‌گیرد.
و مطابق با تغییرات فایل NLU این فایل نیز تغییراتی به جهت قرارگیری intentها خواهد داشت.

3. Stories

```python
def generate_rules_multi_intent(dataframe, file_path):
    with open(file_path, "w", encoding="utf-8") as file:
        file.write('version: "3.1"\n\n')
        file.write("rules:\n")

        for index, record in dataframe.iterrows():
            file.write(f"    - rule: response to Q{record[0]}\n")
            file.write("      steps:\n")
            file.write(f"      - intent: Q{record[0]}\n")
            file.write(f'      - action: utter_Q{record[0]}\n')
```

4. Rules

```python
def generate_stories_multi_intent(dataframe, file_path):
    with open(file_path, "w", encoding="utf-8") as file:
        file.write('version: "3.1"\n\n')
```

بخش ۲ : انجام تنظیمات و آموزش ربات

Sub-intent

تمام تنظیمات مورد نیاز برای آموزش مدل در فایل config باید اعمال شود و پس از آماده سازی دیتاست و این فایل مدل می‌تواند آموزش ببیند با استفاده از کامند rasa train.
فایل config شامل بخش‌های مختلف است که در زیر به توضیح هر یک می‌پردازیم:
در ابتدا باید زبان مورد آموزش مشخص شود که در اینجا فارسی است و به فرم fa نوشته می‌شود.
در بخش بعدی فایل config، مورد pipline قرار دارد که وظیفه پیش‌بینی‌های NLU را برعهده دارد.

در پلتفرم Rasa، هنگامی که پیغام‌های ورودی به سیستم می‌رسند، ابتدا از طریق یک پایپلاین مشخص شده، به صورت ترتیبی پردازش می‌شوند. این پایپلاین می‌تواند شامل چندین قسمت باشد که هرکدام وظایف مختلفی را برعهده دارند. انتخاب یک پایپلاین مناسب برای دیتاست و وظیفه مورد نظر، امری ضروری و حیاتی است.

هر بخش از پایپلاین، پس از دریافت پیغام ورودی، به ترتیبی که در پایپلاین مشخص شده است، اقدام به پردازش پیغام می‌کند. این پردازش می‌تواند شامل عملیاتی مانند تبدیل متن به بردارهای ویژگی (مانند بردارهای tf-idf یا ویژگی‌های تعبیه شده)، پیش‌پردازش متن (مانند حذف توکن‌های متعارف یا نرمال‌سازی)، استخراج ویژگی‌های مهم (مانند شناسایی اجزای معنایی) و سایر عملیات مرتبط با پردازش متن و NLP (پردازش زبان طبیعی) باشد.

انتخاب یک پایپلاین مناسب بسیار مهم است زیرا تعیین می‌کند که چه نوع پردازش‌ها و قابلیت‌هایی در پاسخ به پیغام ورودی انجام می‌شود. هر پایپلاین می‌تواند الگوریتم‌ها و روش‌های مختلفی را برای پردازش متن و استخراج اطلاعات مورد استفاده قرار دهد. بنابراین، تنظیم پایپلاین به‌گونه‌ای که به دیتاست و وظیفه خاصی که سیستم برای آن آموزش داده شده است، مناسب باشد، اهمیت بالایی دارد.

در بخش نخست می‌توان tokenizer مدنظر خود را مشخص نمود. WhitespaceTokenizer این مرحله از پیش پردازش، وظیفه جدا کردن جملات به توکن‌های واژگان را بر عهده دارد. این توکن‌ها به عنوان ورودی به مراحل بعدی می‌روند.

وظیفه فیچرایزر (Featurizer) در یادگیری ماشینی این است که متن ورودی را به بردارهای ویژگی تبدیل کند. این تبدیل می‌تواند به دو شکل صورت گیرد: فیچرهای دنباله (Sequence Features) و فیچرهای جمله (Sentence Features). در فیچرهای دنباله، هر توکن ورودی دارای یک بردار ویژگی است و این امکان را به ما می‌دهد که مدل‌های دنباله‌ای را آموزش دهیم. اما در فیچرهای جمله، ویژگی‌ها و اطلاعات کل جمله در یک بردار ذخیره می‌شوند و می‌توان از آنها در هر مدل حقیبه کلمات (Bag-of-Words) استفاده کرد. به این ترتیب، فیچرایزر نقش مهمی در استخراج و استفاده از ویژگی‌های متنی برای مدل‌های یادگیری ماشینی دارد. که در اینجا از pretrained مدل LanguageModelFeaturizer به جهت تبدیل متن ورودی به بردار ویژگی استفاده شده است.
این dense featurizer که ویژگی‌هایی را برای طبقه‌بندی intent و انتخاب پاسخ ایجاد می‌کند. برای استفاده از این featurizer، باید یک tokenizer مشخص شود و سپس نام مدل و وزن‌های مورد استفاده باید ذکر شود. استفاده از fine-tune نیز یک گزینه است که مدل را بر روی دیتاست موجود بهبود می‌بخشد و برای کاربردهای خاص وظیفه مورد استفاده بهتر است زیرا دقت را افزایش می‌دهد بدون از دست دادن قابلیت تعمیم مدل.

ResponseSelector: این مرحله از مدل مکالمه به عنوان یک انتخاب کننده پاسخ عمل می‌کند. از آن برای جستجوی مدل در پاسخ به سؤالات متداول استفاده می‌شود. متغیرهای `epochs` و `constrain_similarities` برای تنظیم تعداد عملکرد و محدودیت شباهت در انتخاب پاسخ‌ها استفاده می‌شوند.

retrieval_intent بیانگر intent مورد نظر برای response های انتخاب شده است. این بخش همچنین شامل متغیرهایی است که می‌توانند ابعاد و سایز لایه‌های مخفی مدل و تعداد و ابعاد لایه‌های transformer را تغییر دهند. عدم تغییر این پارامترها باعث استفاده از مقادیر پیش‌فرض برای آموزش مدل می‌شود.

Policies
- MemoizationPolicy: این سیاست اطلاعات گذشته را برای تصمیم‌گیری در مورد پاسخ‌ها به یک گفتگو استفاده می‌کند. اگر یک گفتگوی قبلی با الگوی مشابه وجود داشته باشد، این سیاست پاسخ قبلی را به عنوان پاسخ فعلی برمی‌گرداند.
- TEDPolicy: این سیاست از الگوریتم TED (Transformer Embedding Dialogue) برای پیش‌بینی پاسخ‌ها در مکالمات استفاده می‌کند. متغیرهای `max_history` و `epochs` برای تنظیم حافظه گذشته مکالمه و تعداد دوره‌های آموزش مدل TED استفاده می‌شوند.
- RulePolicy: این سیاست برای تعیین قوانین خاصی استفاده می‌شود که با استفاده از الگوهایی تعریف می‌شوند. این سیاست می‌تواند برای تعیین پاسخ‌های ثابت و خاص مورد استفاده قرار گیرد.

در پایان هر فایل config، یک شناسه منحصر به فرد به نام assistant_id قرار داده می‌شود تا ربات‌های مختلف را از یکدیگر تمایز دهد. این شناسه به تمام اطلاعات فنی و مدل‌های استفاده شده در ربات منتقل می‌شود. در صورتی که فایل config شامل این شناسه نباشد، یک نام تصادفی تولید شده و به انتهای فایل config اضافه خواهد شد. با این کار، از همپوشانی و تداخل بین ربات‌ها جلوگیری می‌شود و هر ربات به طور منحصر به فرد شناسایی می‌شود.

Multi-intent

مهمترین تغییر در فایل config برای آموزش مدل با داده‌های multi-intent این است که بخش DIETClassifier را اضافه می‌کنیم.
DIETClassifier یک معماری multi-task است و مخفف Dual Intent and Entity Transformer است که به هدف طبقه‌بندی intent و شناسایی entity در chatbotها مورد استفاده قرار می‌گیرد و بر اساس یک معماری شبکه عصبی است انعطاف پذیری این مدل به جهت قابلیت استفاده از embedingهای pretrained از قبیل ConveRT، GloVe، BERT و غیره را فراهم می کند.

معماری DIETClassifier را می‌توان به دو بخش اصلی تقسیم کرد:
مؤلفه طبقه بندی intent و مؤلفه شناسایی موجودیت (entity)

1. Intent Classification:

مؤلفه تشخیص intent از یک معماری شبکه عصبی مبتنی بر ترانسفورمر برای دسته‌بندی اهداف کاربر استفاده می‌کند. این مؤلفه متن ورودی کاربر را دریافت کرده و آن را از طریق یک sequence of encoding layers پردازش می‌کند تا اطلاعات contextual را ثبت کند. معماری ترانسفورمر به مدل امکان می‌دهد تا به طور موثر وابستگی‌های long-range در متن ورودی را ثبت کند. خروجی از مؤلفه تشخیص هدف یک توزیع احتمال بر روی intentهای از پیش‌تعیین شده است.

2. Entity Recognition:

entity recognition در ماژول DIETClassifier مسئول شناسایی و استخراج موجودیت‌ها از متن ورودی کاربر هستند. این ماژول همچنین از یک معماری شبکه عصبی مبتنی بر Transformer استفاده می‌کند. مدل متن ورودی را پردازش کرده و موقعیت شروع و پایان موجودیت‌ها را در متن پیش‌بینی می‌کند. خروجی entity recognition، یک لیست از موجودیت‌ها همراه با برچسب‌های مربوط به آن‌هاست.

پارامترهای قابل تنظیم در این معماری به شرح زیر است:

Parameter	default	Description
epochs	300	
hidden_layers_sizes		the number of feed-forward layers and their output dimensions for user messages and intents
embedding_dimension	20	output dimension of the embedding layers
number_of_transformer_layers	2	
transformer_size	256	the number of units in the transformer
number_of_attention_heads	4	Number of attention heads in transformer

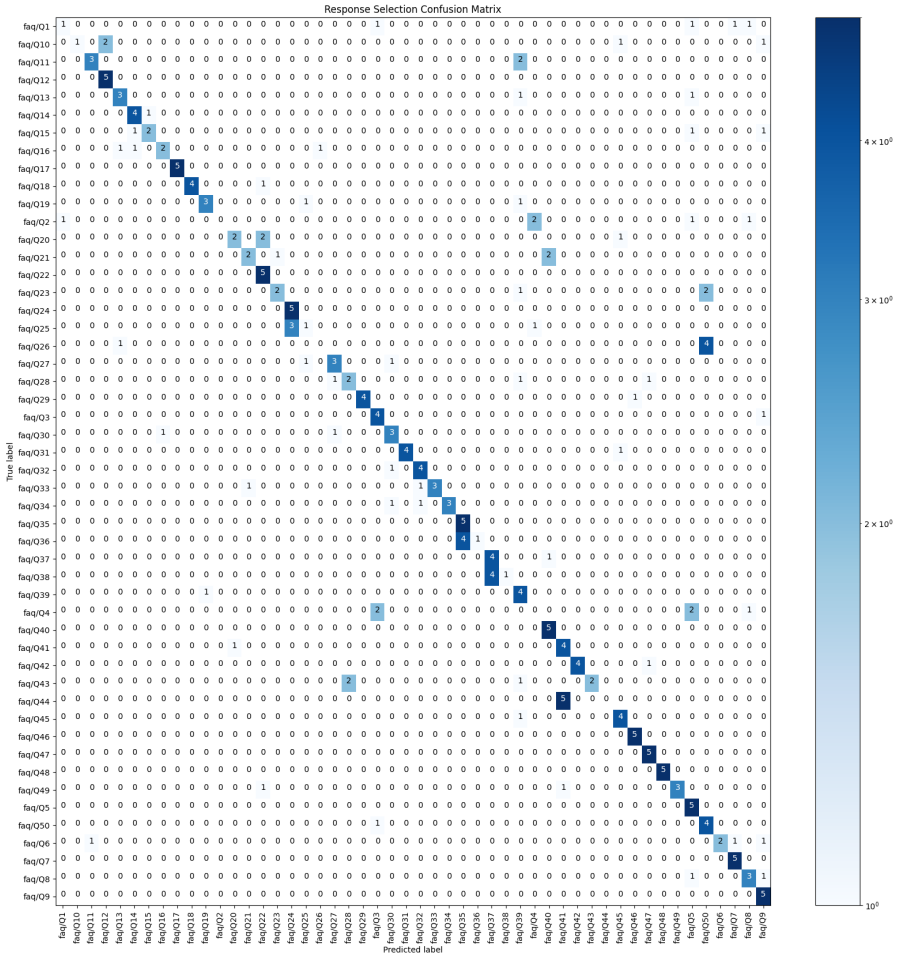
بسته به کاربرد و داده‌ها، پارامترهای دیگر مانند نرخ یادگیری، اندازه دسته و تکنیک‌های منظم نیز می توانند برای بهینه سازی عملکرد مدل تنظیم شوند.

مابقی تنظیمات شبیه به sub-intent است.

سوال ۳: ارزیابی ربات

Sub-intent

Sub-intent - ParsBert with 50 epochs




```
'Accuracy': 0.624
'Precision': 0.653
'Recall': 0.624
'F1 Score': 0.591
```

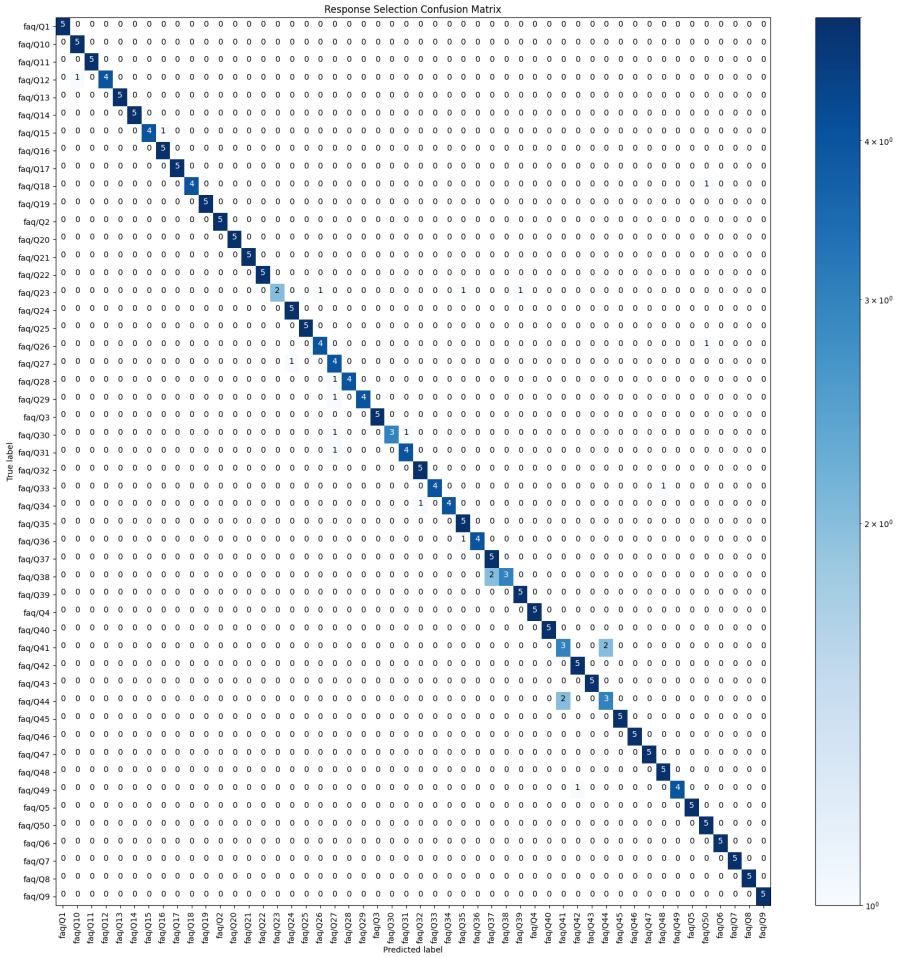
```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "HooshvareLab/bert-base-parsbert-uncased"
  - name: ResponseSelector
    epochs: 50
    retrieval_intent: faq
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 50
    constrain_similarities: true
  - name: RulePolicy
```

دقت مدل با ۵۰ ایپاک برای داده‌های تست قابل قبول است، در اینجا به بررسی چندی از مواردی که به اشتباه دسته‌بندی شده است می‌پردازیم:

- سوالات ۱۰ و ۱۲ در دو مورد به اشتباه دسته بندی شده است که علت آن این است که موضوعات مشابهی را هر دو این سوالات دنبال می‌کنند اولی به بررسی مشکل مرورگر برای ورود به سایت و دومی به بررسی مشکل در ورود به مرورگر به وسیله گوشی سامسونگ می‌پردازد.
- سوالات ۱۱ و ۳۹ در سه مورد اشتباه شده که مجدد برای مشابهت موضوع بوده است اولی برای محدودیت گرفتن اینترنت همراه و دومی برای رزرو اینترنت همراه است.
- سوال ۲۶ کلا درست دسته‌بندی نشده است که علت آن این است که خرید اینترنت اورلپ زیادی با مابقی سوالات دارد.
- سوالات ۳ و ۴ مربوط به مشکلات gprs است که یکی به بررسی کد دستوری و دومی از طریق پیامک می‌پردازد.
- سوالات ۲۴و ۲۵ اولی به بررسی اینکه نوترینو چیست می‌پردازد و دومی به تفاورت بسته و سرویس نوترینو می‌پردازد که بسیار شبیه به هم اند.
- سوالات ۴۳ و ۲۸: که به تمدید خودکار بسته‌ها یکی بعد از زمان مشخص و دیگری بعد از اتمام بسته‌های مختلف می‌پردازد.

در حالت کلی این سوالات که به اشتباه دسته‌بندی شده‌اند بسیار مشابه بوده و برای آنکه مدل در این موارد هم به درستی عمل کند نیازاست که بردارهای ویژگی بامعنایی استخراج شده تا دسته‌بندی درست رخ دهد یا دسته‌بند قوی‌تری مورد استفاده قرار گیرد.

Sub-intent - LabSE with 50 epochs



'Accuracy': 0.908
'Precision': 0.925
'Recall': 0.908
'F1 Score': 0.907

```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "rasa/LaBSE"
  - name: ResponseSelector
    epochs: 50
    retrieval_intent: faq
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 50
    constrain_similarities: true
  - name: RulePolicy
```

با به کار گیری labse دقت مدل حدود ۳۰ درصد افزایش پیدا کرد نسبت به parsbert.
حال به بررسی موارد مهمی که به اشتباه دسته‌بندی شده است می‌پردازیم:

- سوالات ۱۲و ۱۰: در دو مورد به اشتباه دسته بندی شده است که علت آن این است که موضوعات مشابهی را هر دو این سوالات دنبال می‌کنند اولی به بررسی مشکل مرورگر برای ورود به سایت و دومی به بررسی مشکل در ورود به مرورگر به وسیله گوشی سامسونگ می‌پردازد.

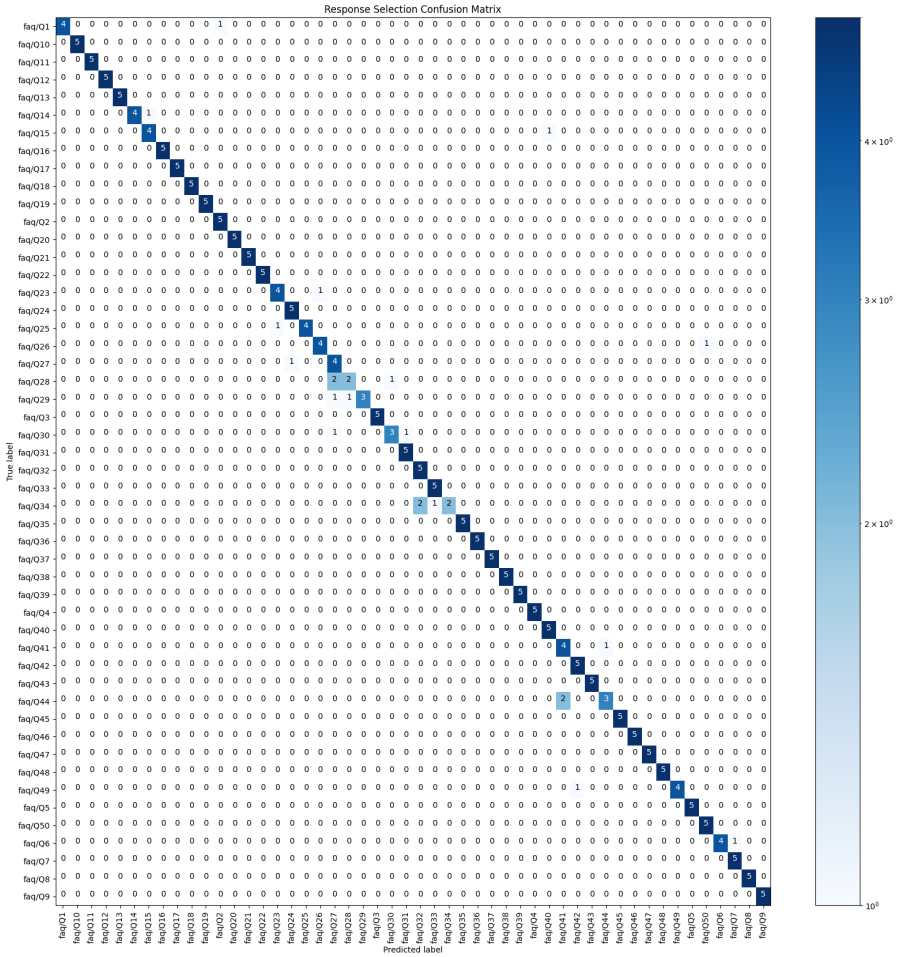
- سوالات ۱۵ و ۱۶: اخبار مربوط به نسل ۴ اینترنت و دومی پیرامون اینکه چگونه می‌توان از نسل ۴ اینترنت استفاده نمود.
- سوال ۲۷: درمورد بسته‌های ساعتی همراه اول است که اورلپ زیادی با مابقی سوالات دارد به علت جنرال بودن سوال.
- سوالات ۳۷ و ۳۸: موضوع مربوط به این دو سوال بسیار مشابه بوده و در موارد جزئی مثل نوع بسته تفاوت دارند.
- **سوالات ۴۱ و ۴۴: اولی طریقه اطلاع رسانی پیرامون بسته رزرو است و دومی برای استعلام بسته‌های رزرو بوده است.**

مدل LaBSE معمولا به عنوان یک مدل چندزبانه (multi-lingual) استفاده می‌شود. این به این معنی است که این مدل با توجه به ویژگی‌های مشترک بین زبان‌ها، توانایی درک و تفسیر داده‌ها در زبان‌های مختلف را دارد. در صورتی که مدل LaBSE با دادگان بیشتری آموزش ببیند، احتمالا قدرتش در درک و تفسیر داده‌ها افزایش خواهد یافت.

همچنین، به علت محدودیت تعداد توکن‌های موجود برای زبان فارسی در مدل LaBSE، فرآیند توکن‌بندی (tokenization) ممکن است به صورت نزدیک‌تر به سطح کاراکتر انجام شود. این موضوع می‌تواند یکی از دلایل بهبود قابل توجه عملکرد مدل باشد.

برای افزایش دقت مدل، می‌توان آن را بیشتر آموزش داد. به عنوان مثال، می‌توان مدل LaBSE را به مدت ۱۰۰ و ۲۰۰ دور آموزش داد. نتایج این آموزش به ترتیب ممکن است به صورت زیر باشد.

Sub-intent - LabSE with 100 epochs

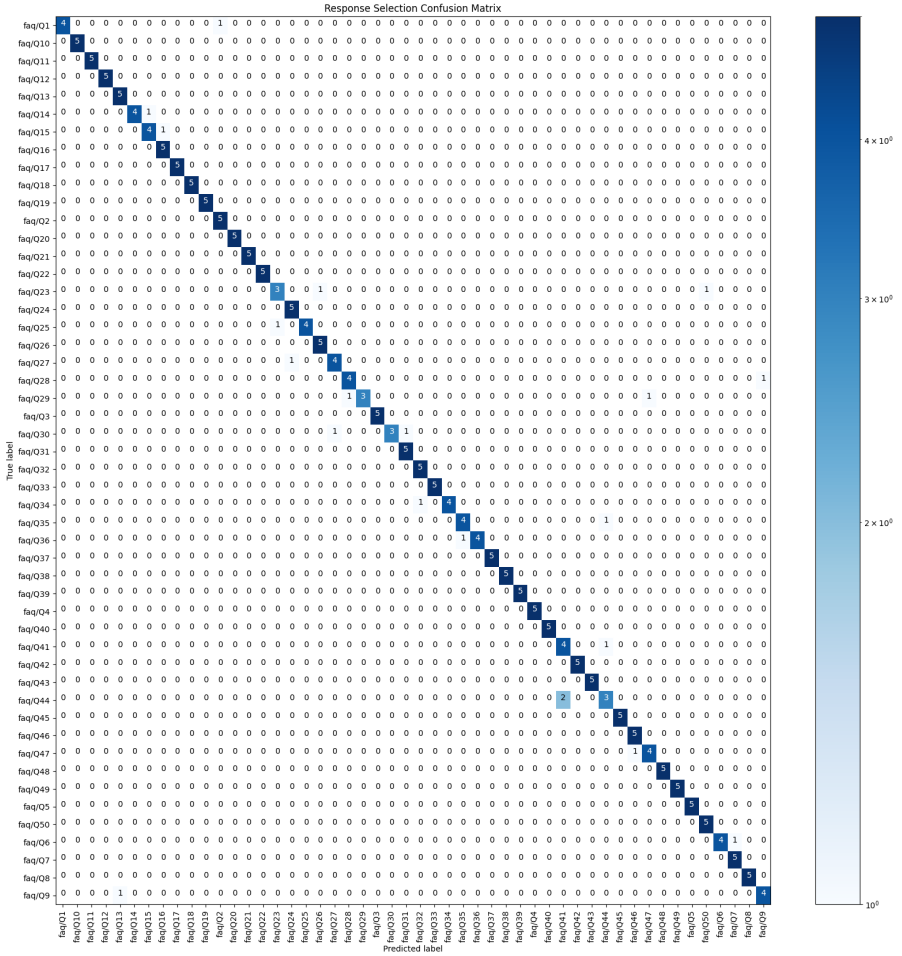



```
'Accuracy': 0.912
'Precision': 0.922
'Recall': 0.912
'F1 Score': 0.908
```

```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "rasa/LaBSE"
  - name: ResponseSelector
    epochs: 100
    retrieval_intent: faq
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
    constrain_similarities: true
  - name: RulePolicy
```

اضافه کردن ۵۰ اپیاک به مدل قبلی تنها ۰.۰۰۱ درصد با افزایش دقت مدل اضافه نمود و تاثیر زیادی در قوی‌تر شدن مدل نداشت.
مابقی نتایج و تحلیل‌ها شباهت به بخش قبل دارد و مجدد بیان نمی‌شود.

Sub-intent - LabSE with 200 epochs




```
'Accuracy': 0.916
'Precision': 0.923
'Recall': 0.916
'F1 Score': 0.914
```

```yaml
language: fa
pipeline:
- name: WhitespaceTokenizer
- name: LanguageModelFeaturizer
  model_name: "bert"
  model_weights: "rasa/LaBSE"
- name: ResponseSelector
  epochs: 200
  retrieval_intent: faq
  constrain_similarities: true
  cache_dir: null
policies:
- name: MemoizationPolicy
- name: TEDPolicy
  max_history: 5
  epochs: 200
  constrain_similarities: true
- name: RulePolicy
assistant_id: 20230701-080733-lenient-moscato
```

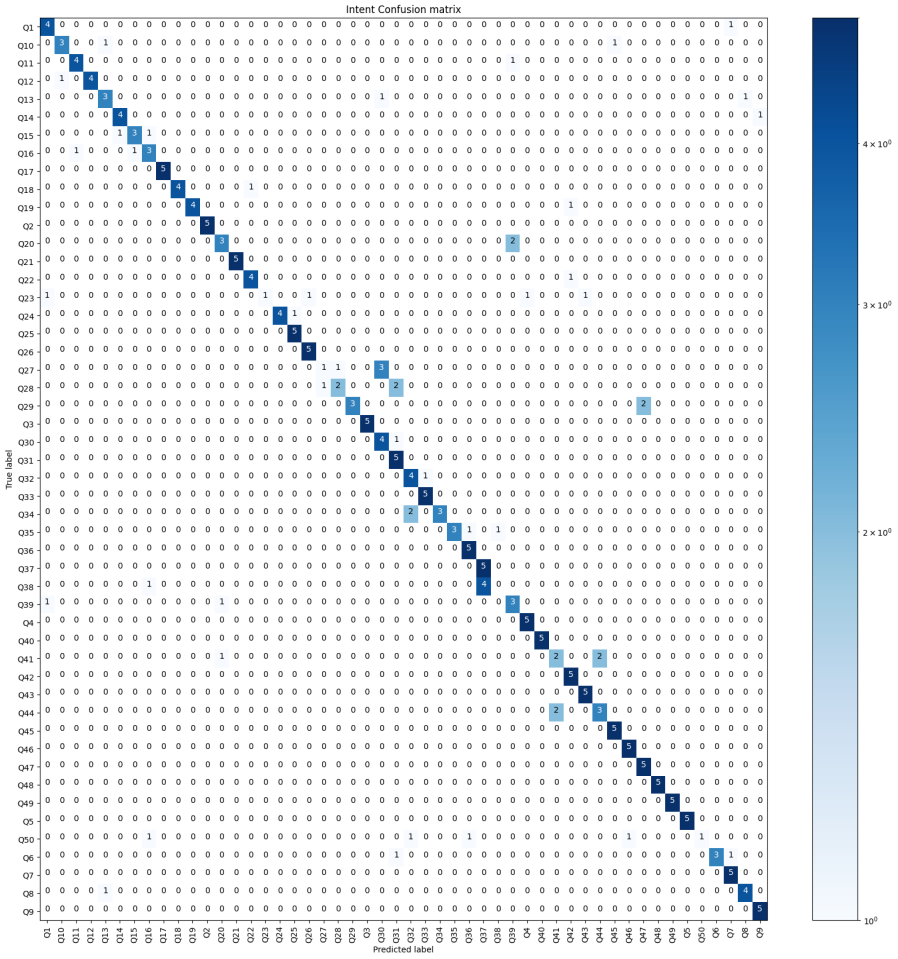
افزایش حدود یک درصدی f1 نشان می‌دهد که با افزایش ۱۰۰تایی میزان اپاک‌هایمان مدل قوی‌تری می‌رسیم درنتیجه هنوز مدل overfit نشده است و شاید با افزایش اپاک باز هم به مدل دقیق‌تری برسیم.

حال به بررسی موارد اشتباه دسته‌بندی شده می‌پردازیم.

- سوالات ۴۵ و ۴۴ بازرترین اشتباه مدل است: **اولی طریقه اطلاع رسانی پیرامون بسته رزرو است و دومی برای استعلام بسته‌های رزرو بوده است.**
- مابقی سوالات اشتباهاتی در حد یک سوال داشته‌اند که بسیار تعداد کمی است.

Multi-intent

Multi-intent - ParsBert with 50 epochs



'Accuracy': 0.776
'Precision': 0.79
'Recall': 0.776
'F1 Score': 0.758

```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "HooshvareLab/bert-base-parsbert-uncased"
  - name: ResponseSelector
    epochs: 50
    constrain_similarities: true
    cache_dir: null
  - name: DIETClassifier
    epochs: 50
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    constrain_similarities: true
  - name: RulePolicy
```

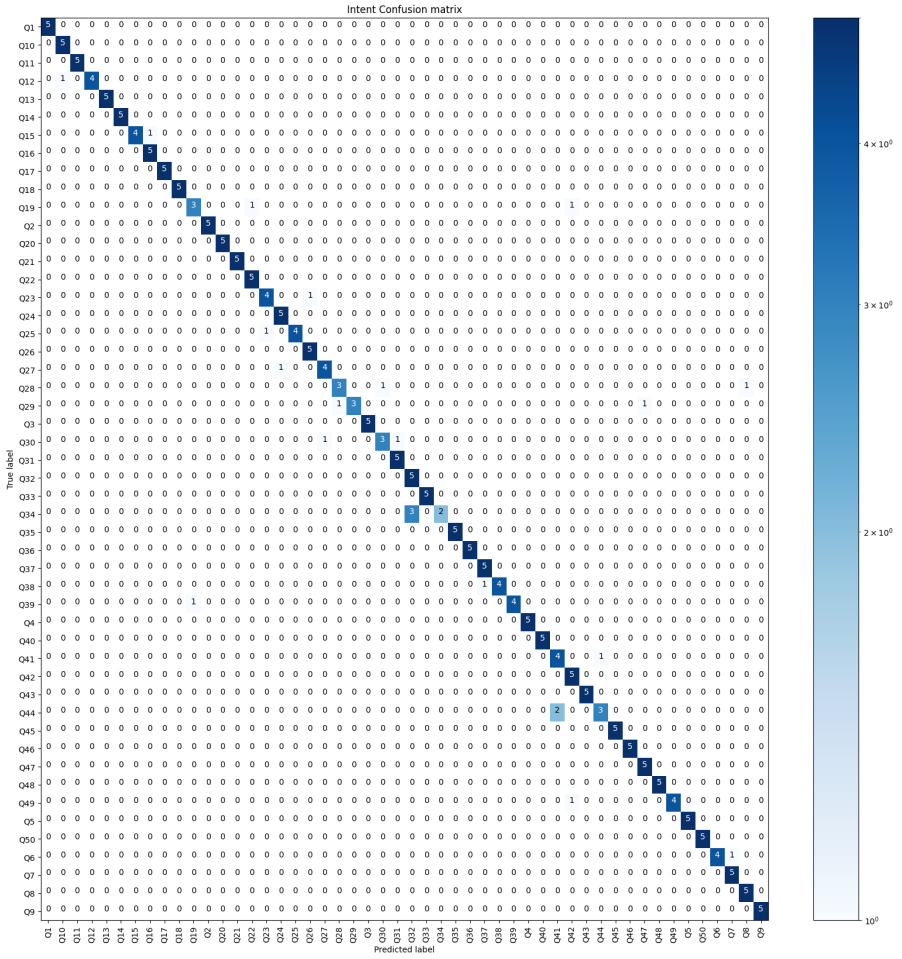
حالت multi-intent به وسیله parsbert در مقایسه با با تعداد اپیاک مشابه حدود ده درصد افزایش دقت داشته است.
استفاده از الگوریتم DIETClassifier در شناسایی intentها، نسبت به تشخیص response برای یک کلاس intent، آسان‌تر است. این الگوریتم می‌تواند به شبکه‌های پیچیده‌تری با استفاده از DIETClassifier منجر شود. ابتدا، این شبکه، intent مورد نظر را شناسایی می‌کند و سپس با استفاده از ResponseSelector، پاسخ مناسب را بازگردانده می‌شود. همچنین، در مستندات ابزار Rasa، ذکر شده است که ResponseSelector تنها می‌تواند با ویژگی‌های خاصی کار کند و این ویژگی‌ها بهتر از همه در LaBSE استخراج می‌شوند. بنابراین، استفاده از الگوریتم DIETClassifier به همراه مدل ParsBERT منجر به افزایش دقت چت‌بات می‌شود.

مواردی که به اشتباه دسته‌بندی شده است به شرح زیر است:

- سوالات ۲۰ و ۳۹: رزرو اینترنت همراه نقطه مشترک بین این دو سوال است.
- سوال ۲۷: مشخصات مربوط به بسته اینترنتی در ساعت مشخص که اورلپ زیادی دارد با مابقی سوالات.
- سوالات ۲۸ و ۳۱: بسته‌های ساعت مشخص.
- سوالات ۴۱ و ۴۴: **اولی طریقه اطلاع رسانی پیرامون بسته رزرو است و دومی برای استعلام بسته‌های رزرو بوده است.**
- سوالات ۲۹ و ۴۷: خرید بسته اینترنتی.

توجه به این نکته ضروری است که سوالاتی که در multi-intent به اشتباه دسته‌بندی شده است نسبت به sub-intent بسیار متفاوت است و می‌توان گفت این دو رویکرد دیدگاه مدل را نسبت به جملات تغییر می‌دهد و از دید دیگری به ماجرا نگاه می‌کند. اما سوالاتی مثل ۴۱ و ۴۴ در هر دو رویکرد اشتباه طبقه‌بندی شده است.

Multi-intent - LabSE with 50 epochs

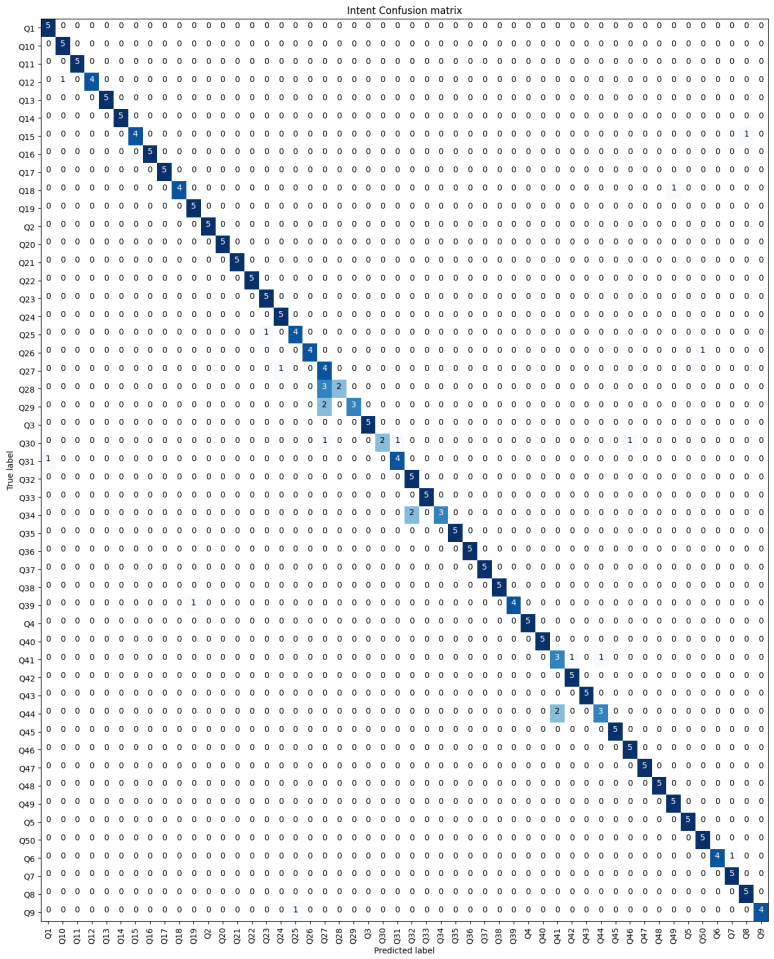


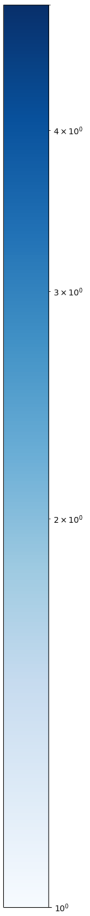
'Accuracy': 0.908
'Precision': 0.918
'Recall': 0.908
'F1 Score': 0.904

```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "rasa/LaBSE"
  - name: ResponseSelector
    epochs: 50
    constrain_similarities: true
    cache_dir: null
  - name: DIETClassifier
    epochs: 50
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    constrain_similarities: true
  - name: RulePolicy
```

استفاده از labse منجر به افزایش دقت حدود ۲۰ درصدی می‌شود.
در مقایسه این مدل با مدل مشابه در sub-intent، sub-intent دقت بالاتری در حد ۰.۰۰۳ دارد.
در این مدل سوالات ۳۴، ۳۲ و ۴۴ و ۴۱ به اشتباه طبقه بندی شده است.

Multi-intent - LabSE with 100 epochs



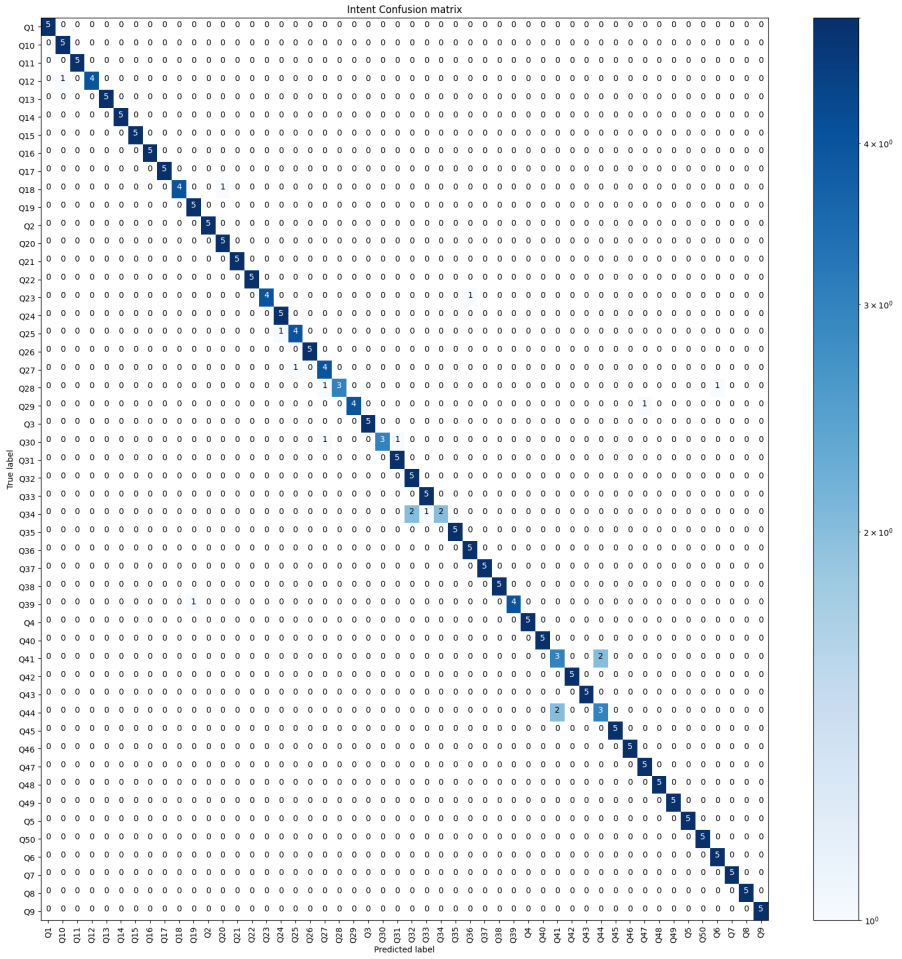


```
'Accuracy': 0.904
'Precision': 0.924
'Recall': 0.904
'F1 Score': 0.901
```

```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "rasa/LaBSE"
  - name: ResponseSelector
    epochs: 100
    constrain_similarities: true
    cache_dir: null
  - name: DIETClassifier
    epochs: 100
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    constrain_similarities: true
  - name: RulePolicy
```

در این مدل نسبت به ۵۰ اپیاک کمتر دقت کاهش یافته و در مقایسه با مدل sub-intent نیز بدتر عمل کرده است.

Multi-intent - LabSE with 200 epochs




```
'Accuracy': 0.928
'Precision': 0.937
'Recall': 0.928
'F1 Score': 0.925
```

```yaml
language: fa
pipeline:
  - name: WhitespaceTokenizer
  - name: LanguageModelFeaturizer
    model_name: "bert"
    model_weights: "rasa/LaBSE"
  - name: ResponseSelector
    epochs: 200
    constrain_similarities: true
    cache_dir: null
  - name: DIETClassifier
    epochs: 200
    constrain_similarities: true
    cache_dir: null
policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 5
    constrain_similarities: true
  - name: RulePolicy
```

بهترین عملکرد تمامی مدل‌ها را مدل فعلی با f1 حدود ۹۲ درصد دارد.

و سولات ۴۱و ۴۴ همچنین سوالات ۳۴و ۳۲ به اشتباه طبقه‌بندی شده است.

مقایسه نهایی

- همان‌طور که مشاهده می‌شود مدل parsbert در مقایسه با labse در تمامی حالات بهتر عمل کرده است.
- در اکثر مدل‌ها با افزایش تعداد ایپاک دقت افزایش پیدا کرده است.
- مدل multi-intent در حالت labse و sub-intent در حالت labse تفاوت چندانی با هم نداشته‌اند.
- بهترین دقت به دست آمده ۹۲ درصد بوده است که مربوط به multi-intent with labse بوده است.

بخش دوم – استخراج مقادیر ارزش‌ها