به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس پردازش زبان‌های طبیعی

تمرین شش

نام و نام خانوادگی	محمد جواد رنجبر
شماره دانشجویی	۸۱۰۱۰۱۱۷۳
تاریخ ارسال گزارش	۱۴۰۲.۰۴.۰۶

فهرست

شکل‌ها

پاسخ ۱. ربات پاسخگو به پرسش‌های پرتکرار

آماده‌سازی داده‌ها به صورت دستی

می‌توان به صورت دستی داده‌ها را در فایل‌های nlu.yml و domain.yml و rule.yml قرار داد، اما اینکار پرزحمت است، پس با استفاده از کدی که در بخش بعدی توضیح خواهد داده شد این کار انجام می‌شود.

آماده‌سازی داده‌ها به صورت خودکار

توسط کد داخل cov.py انجام شده است.

رویکرد ۱:

به این صورت عمل می‌کنیم که فایل‌های xsxl را باز کرده و ما دارای ۵۰ intent مختلف هستیم. این intentها هرکدام دارای ۱۵ سوال متداول(Frequently asked questions)[1] هستند. لذا این سوال‌ها را در فایل nlu.yml کپی می‌کنیم. همچنین نام این intentها را faq گذاشته و domainها موردنیاز را نیز تولید می‌کنیم و پاسخ مربوط به این intentها را نیز در فایل domain.yml قرار می‌دهیم.

در فایل rule.yml نیز faq و پاسخ مربوط به آن را مشخص می‌کنیم.

رویکرد ۲:

در این رویکرد سه کلاس تعریف کرده‌ایم که یک کلاس مربوط به پرسش‌های پرتکرار می‌باشد و دو کلاس دیگر مربوط به خوش‌آمدگویی و خداحافظی است. در هر کلاس نیز subintentهای مختلفی تعریف شده است. برای این کار از کلمه fag/# استفاده کرده‌ایم که نشان‌دهنده شماره subintent می‌باشد و در nlu.yml سوالات ما قرار داده شده است، همچنین جملاتی در مورد خوش‌امدگویی و خداحافظی از طرف کاربر نیز در intentهای welcome و bye قرار داده شده است.

در domain.yml نیز پاسخ مربوط به intentها و subintentهای مختلف قرار داده شده است، برای پرسش‌های پرتکرار پاسخ مربوط داخل مجموعه‌داده قرار داده است و برای welcome و bye نیز چند پاسخ مختلف قرار داده شده است که ربات به صورت تصادفی یکی را به کاربر می‌دهد.

```
  utter_faq/50:
  - text: " را شمارەگیر  *10*100#و برای استعلام کد  *332*100#برای فعالسازی بسته های نامحدود شبانه کد
ی نمایید."

  utter_welcome:
```

[1] FAQ

```
  - text: "سلام! چه خدمتی ازم برمیاد؟"
  - text: "سلام! روزتون بخیر"
  - text: "سلام!"

  utter_bye:
  - text: "روز خوبی داشته باشید."
  - text: "خدانگهدار"
  - text: "خداحافظ"
```

در rule.yml نیز intentها و پاسخ‌های مربوط به آن‌ها مشخص شده است.

```
rules:

- rule: faq
  steps:
  - intent: faq
  - action: utter_faq

- rule: welcome
  steps:
  - intent: welcome
  - action: utter_welcome

- rule: bye
  steps:
  - intent: bye
  - action: utter_bye
```

انجام تنظیمات و آموزش ربات

برای تنظیمات ربات از فایل config.yml استفاده می‌کنیم. برای اینکار ابتدا یک بار از فرمان rasa init استفاده می‌کنیم تا این فایل با مقادیر اولیه تولید شود حال وابسته به درخواست سوال و نیاز خود این فایل را تغییر خواهیم داد.

اجزای config.yml:

Recipe: مربوط به configهای مختلف و معماری‌های مدل مختلف است که در حال حاضر دو نوع "default.v1" و "graph.v1" در دسترس هستند و ما از default.v1 استفاده کردیم.

assistant_id: برای این استفاده می‌شود که تعدادی از دستیاران که در حال توسعه هستند از همدیگر قابل تشخیص باشند.

language: مشخص کننده زبان مدل برای آموزش است.

Pipeline: نشان‌دهنده شروع برای pipline پروسه‌های مدل NLU ما می‌باشد.

Whitespacetokenizer: این بخش بر اساس فواصل (space) ورودی را tokenize می‌کند.

RegexFeaturizer: این بخش بر اساس منطق عبارات منظم از جمله ویژگی استخراج می‌کند.
LexicalSyntacticFeaturizer: این بخش بر اساس ویژگی‌های واژگانی و نحوی را از متن استخراج می‌کند.
CountVectorsFeaturizer: این بخش بر اساس فراکانس کلمات ویژگی استخراج می‌کند.

- analyzer: این خط شمارنده کاراکترها را مشخص کرده‌ایم که char_wb می‌باشد.
- Min_ngram: این خط حداقل طول کاراکتر n-gram را برای مشخص کننده شمارش بردار ویژگی‌ها می‌باشد.
- max_ngram: این خط حداکثر طول کاراکتر n-gram را برای مشخص کننده شمارش بردار ویژگی‌ها می‌باشد.

LanguageModelFeaturizer: با استفاده از یک مدل زبانی از پیش آموزش‌دیده، بازنمایی‌های کلمه‌ای متنی[1] را استخراج می‌کند.
model_name: مشخص کننده نام مدل مورد استفاده خواهد بود که اینجا از مدل Bert استفاده کرده‌ایم.
model_weights: در اینجا مشخص می‌کنیم وزن اولیه مدل ما چه باشد، که در این پروژه از دو مدل ParsBert و LaBSE استفاده کرده‌ایم.
DIETClassifier: یک Dialogue Intention Entity Transformer classifier می‌باشد که برای طبقه‌بندی intentها استفاده می‌شود.

- Epochs: که مشخص‌کننده تعداد دفعات آموزش مدل است.
- constrain_similarities: این خط شباهت های محدود کننده را در طول آموزش برای طبقه بندی کننده DIET امکان پذیر می کند.

EntitySynonymMapper: این خط یک مؤلفه Entity Synonym Mapper را پیکربندی می کند. Entity Synonym Mapper برای نگاشت تغییرات یا مترادف های مختلف مقادیر موجودیت به یک شکل متعارف رایج استفاده می شود. این به عادی سازی مقادیر موجودیت و بهبود دقت استخراج موجودیت کمک می کند. به عنوان مثال، اگر یک موجودیت "مکان" با تغییراتی مانند "نیویورک"، "نیویورک" و "اپل بزرگ" دارید، Entity Synonym Mapper می تواند همه آنها را به یک شکل متعارف، مانند "شهر نیویورک" نگاشت کند.
برای رویکرد دوم چند بخش به این فایل اضافه می‌شود که به شرح زیر می‌باشد:
ResponseSelector: با استفاده از یک رویکرد مبتنی بر بازیابی[2] کار می کند، جایی که ورودی کاربر را با مجموعه ای از مثال ها یا الگوهای آموزشی از پیش تعریف شده مطابقت می دهد. هر مثال آموزشی شامل یک پرسش کاربر و پاسخ مربوطه است. در طول آموزش، ResponseSelector یاد می گیرد که مناسب ترین پاسخ را برای یک پرس و جوی کاربر معین بر اساس تطبیق شباهت پیش بینی کند.

- Epochs: تعداد دوره آموزش برای ResponseSelector را مشخص می‌کند.

[1] contextualized word representations
[2] Retrieval

- constrain_similarities: این کمک می کند تا اطمینان حاصل شود که پرس و جوهای مشابه پاسخ های مشابهی دریافت می کنند.
- retrieval_intent: faq: این نشان می دهد که مؤلفه باید پاسخ هایی را به طور خاص برای سؤالات متداول بازیابی کند.

برای سوال دو نیز CRFEntityExtractor نیاز است که در اینجا توضیح داده می‌شود:

CRFEntityExtractor: CRFEntityExtractor. برای استخراج entityها از متن ورودی کاربر استفاده می شود. CRFEntityExtractor یک مدل CRF را بر روی داده های آموزش می‌دهد تا برچسب‌های entity را برای متن ورودی داده شده پیش بینی کند.

حال config نهایی به شبیه شکل زیر خواهد بود:

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased
- name: DIETClassifier #indent classifer
  epochs: 50
  constrain_similarities: true
- name: EntitySynonymMapper
policies: null
```

حال مدل‌های خواسته شده را آموزش می‌دهیم، ابتدا مدل‌ها را با ۵۰ ایپاک آموزش می‌دهیم، بهترین مدل بر اساس دقت و خطا مدل LaBSE بود. به نظر که بهترین مدل، LaBSE با ۱۰۰ دوره می‌باشد که با ۵۰ epoch مدل همچنان جا برای بهتر شدن دارد و با ۲۰۰ epoch تغییر خاصی رخ نداده است و احتمالا مدل تا آنجا overfit شده است. به طور کلی با LaBSE نتیجه‌ی بهتری گرفته‌ایم.

آموزش و ارزیابی ربات

رویکرد ۱:

مدل LaBSE با epoch=50:

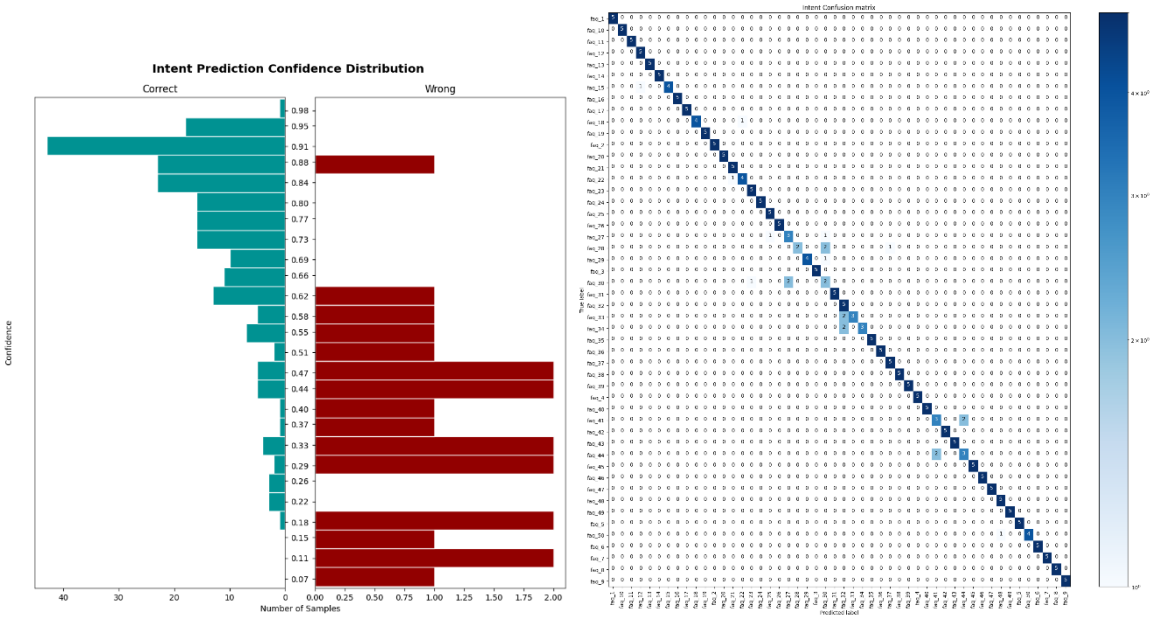
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```yaml
model_weights: rasa/LaBSE
epochs: 50
```

کل config به شکل زیر می‌باشد:

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased
- name: DIETClassifier #indent classifer
  epochs: 50
  constrain_similarities: true
- name: EntitySynonymMapper
policies: null
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۱ عملکرد مدل **LaBSE** با ۵۰ **epoch** و رویکرد ۱

برای intentها مدل به صورت زیر عمل می‌کند:


```
"accuracy": 0.916,
"macro avg": {
  "precision": 0.9297777777777777,
  "recall": 0.9159999999999999,
  "f1-score": 0.9151890331890331,
  "support": 250
},
"weighted avg": {
  "precision": 0.9297777777777777,
  "recall": 0.916,
  "f1-score": 0.9151890331890331,
  "support": 250
},
```

```
  "micro avg": {
    "precision": 0.916,
    "recall": 0.916,
    "f1-score": 0.916,
    "support": 250
  }
```

مدل LaBSE با epoch=100:

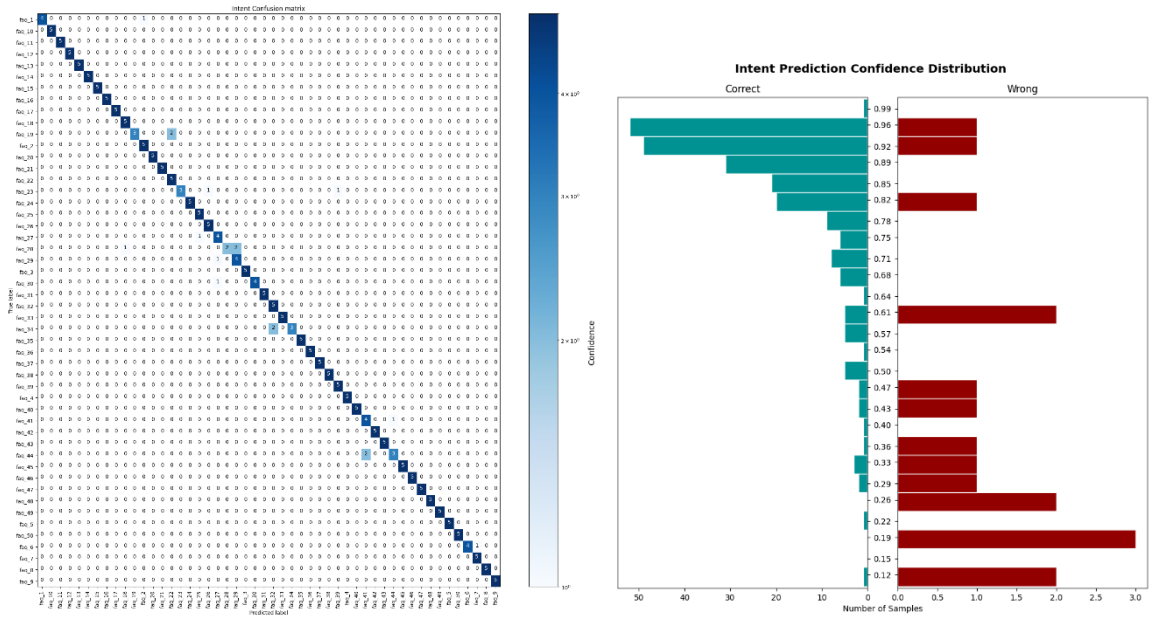
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: rasa/LaBSE
epochs: 100
```

کل config به شکل زیر می‌باشد:

```
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa
pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 100
  constrain_similarities: true
- name: EntitySynonymMapper

policies: null
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۲ عملکرد مدل **LaBSE** با ۱۰۰ **epoch** و رویکرد ۱

برای intentها مدل به صورت زیر عمل می‌کند:


```
  "accuracy": 0.932,
  "macro avg": {
    "precision": 0.9435714285714286,
    "recall": 0.932,
    "f1-score": 0.9291558441558441,
    "support": 250
  },
  "weighted avg": {
    "precision": 0.9435714285714285,
    "recall": 0.932,
    "f1-score": 0.9291558441558442,
    "support": 250
  },
  "micro avg": {
    "precision": 0.932,
    "recall": 0.932,
    "f1-score": 0.932,
    "support": 250
  }
```


مدل LaBSE با epoch=200:

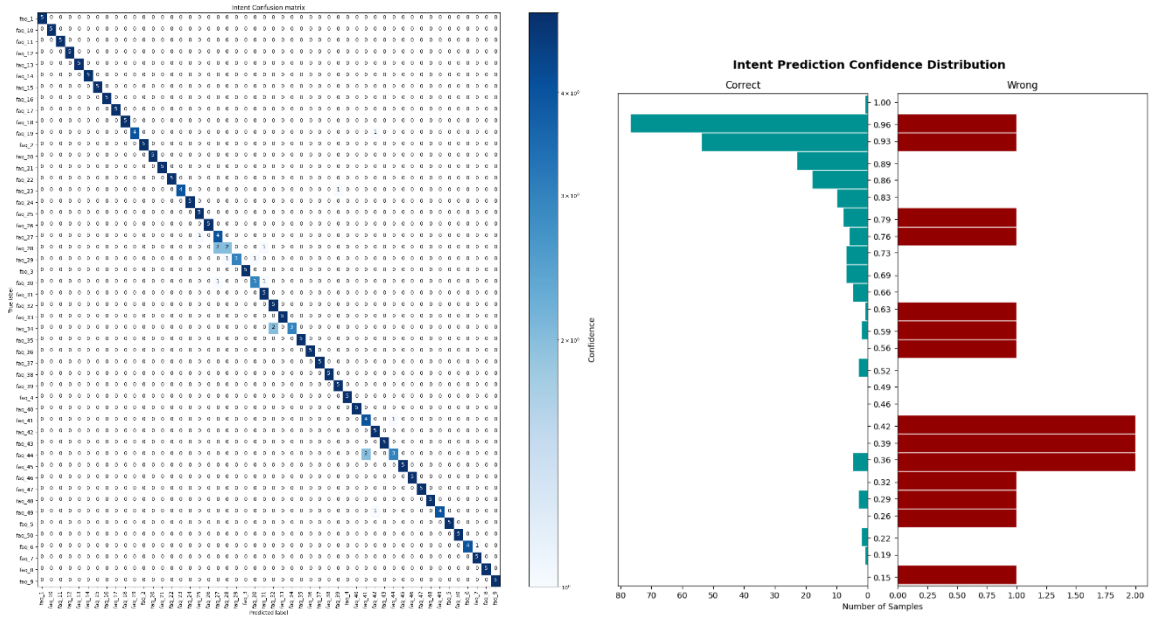
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: rasa/LaBSE
epochs: 200
```

کل config به شکل زیر می‌باشد:

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa
pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 200
  constrain_similarities: true
- name: EntitySynonymMapper
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۳ عملکرد مدل **LaBSE** با ۲۰۰ **epoch** و رویکرد ۱

برای intentها مدل به صورت زیر عمل می‌کند:

```
"accuracy": 0.932,
"macro avg": {
  "precision": 0.9409523809523809,
  "recall": 0.9319999999999999,
  "f1-score": 0.9302020202020203,
  "support": 250
},
"weighted avg": {
  "precision": 0.940952380952381,
  "recall": 0.932,
  "f1-score": 0.9302020202020203,
```

```
    "support": 250
  },
  "micro avg": {
    "precision": 0.932,
    "recall": 0.932,
    "f1-score": 0.932,
    "support": 250
  }
```


مدل ParsBert با epoch=50:

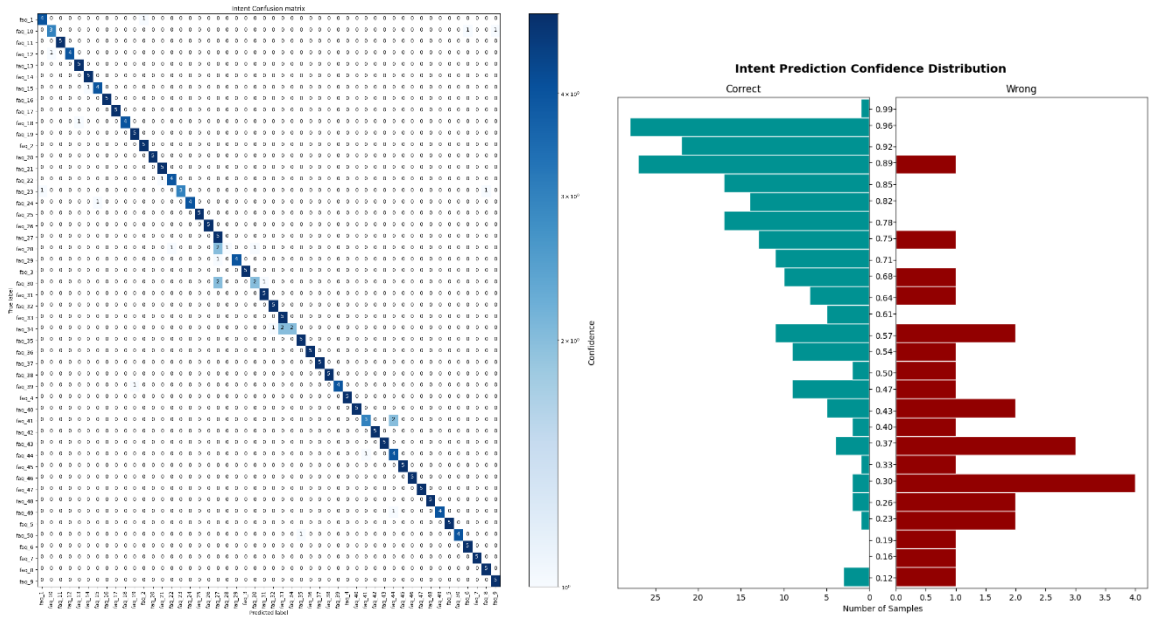
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: HooshvareLab/bert-base-parsbert-uncased
epochs: 50
```

کل config به شکل زیر می‌باشد:

```
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:

- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  #model_weights: rasa/LaBSE
  model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 50
  constrain_similarities: true
- name: EntitySynonymMapper
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۴ عملکرد مدل **ParsBert** با ۵۰ **epoch** و رویکرد ۱

برای intentها مدل به صورت زیر عمل می‌کند:

```
  "accuracy": 0.892,
  "macro avg": {
    "precision": 0.9103809523809524,
    "recall": 0.8919999999999999,
    "f1-score": 0.8855396825396825,
    "support": 250
  },
  "weighted avg": {
    "precision": 0.9103809523809524,
    "recall": 0.892,
    "f1-score": 0.8855396825396824,
    "support": 250
  },
  "micro avg": {
    "precision": 0.892,
    "recall": 0.892,
    "f1-score": 0.892,
    "support": 250
  }
```

مدل ParsBert با epoch=100:

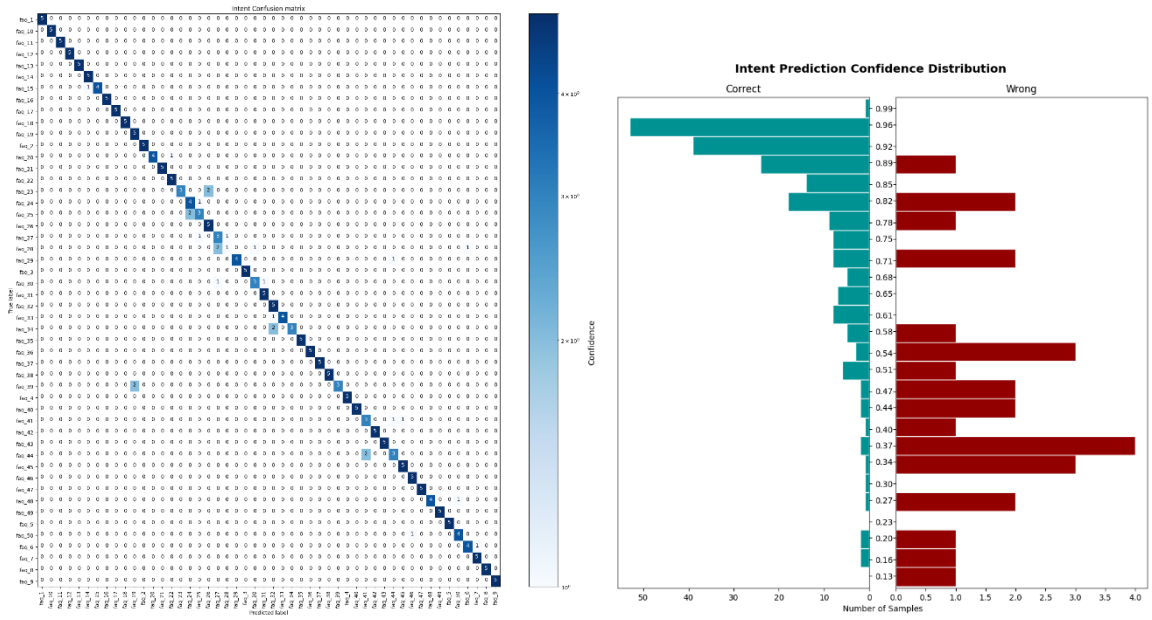
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: HooshvareLab/bert-base-parsbert-uncased
epochs: 100
```

کل config به شکل زیر می‌باشد:

```yaml
recipe: default.v1

assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  #model_weights: rasa/LaBSE
  model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 100
  constrain_similarities: true
- name: EntitySynonymMapper
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۵ عملکرد مدل **ParsBert** با ۱۰۰ **epoch** و رویکرد ۱

برای intentها مدل به صورت زیر عمل می‌کند:

```
"accuracy": 0.888,
"macro avg": {
  "precision": 0.8974047619047617,
  "recall": 0.888,
  "f1-score": 0.8841999111999113,
  "support": 250
},
"weighted avg": {
  "precision": 0.8974047619047619,
  "recall": 0.888,
```

```
    "f1-score": 0.8841999111999111,
    "support": 250
  },
  "micro avg": {
    "precision": 0.888,
    "recall": 0.888,
    "f1-score": 0.888,
    "support": 250
  }
```

مدل ParsBert با epoch=200:

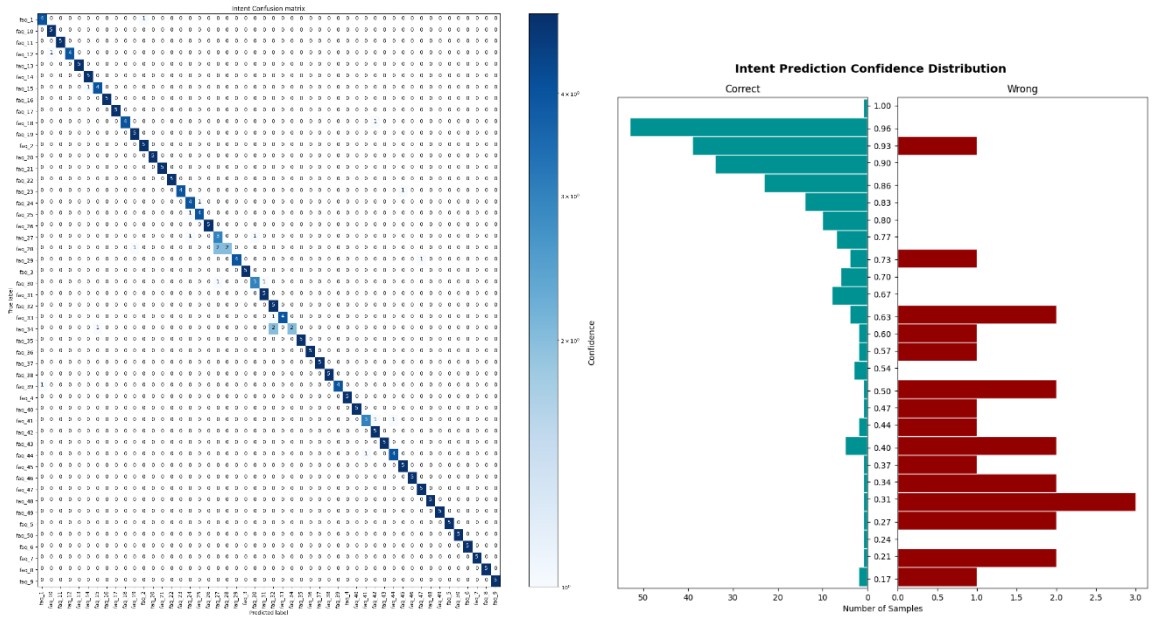
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: HooshvareLab/bert-base-parsbert-uncased
epochs: 200
```

کل config به شکل زیر می‌باشد:

```
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  #model_weights: rasa/LaBSE
  model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 200
  constrain_similarities: true
- name: EntitySynonymMapper
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۶ عملکرد مدل **ParsBert** با ۲۰۰ **epoch** و رویکرد ۱

برای intentها مدل به صورت زیر عمل می‌کند:


```
    "accuracy": 0.908,
  "macro avg": {
    "precision": 0.9207857142857142,
    "recall": 0.9079999999999999,
    "f1-score": 0.9049690309690309,
    "support": 250
  },
  "weighted avg": {
    "precision": 0.9207857142857142,
    "recall": 0.908,
    "f1-score": 0.904969030969031,
    "support": 250
  },
  "micro avg": {
    "precision": 0.908,
    "recall": 0.908,
    "f1-score": 0.908,
    "support": 250
  }
```


رویکرد ۲:

مدل LaBSE با epoch=50:

این مدل را با تغییر دو config زیر آموزش می‌دهیم:

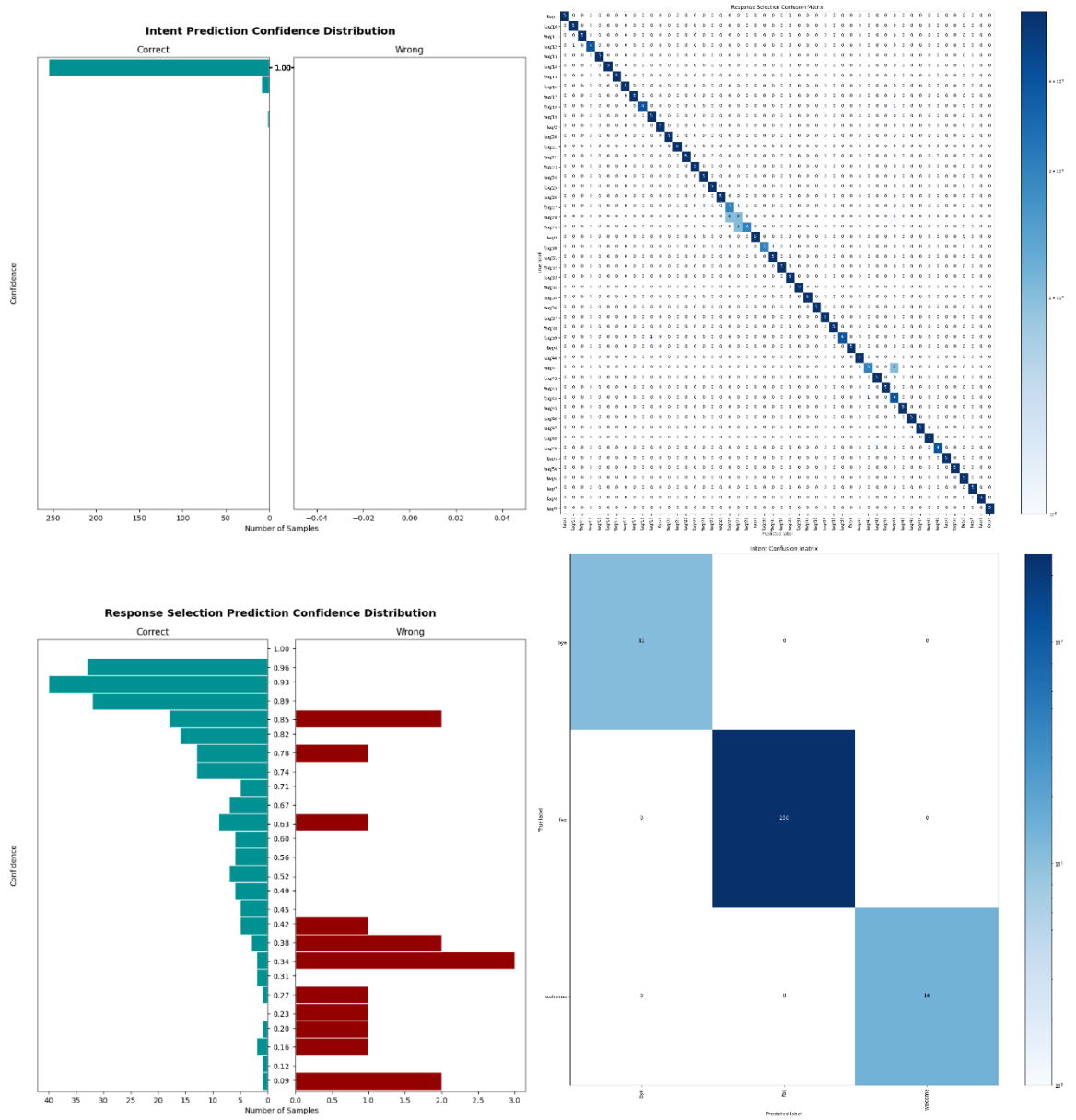
```
model_weights: rasa/LaBSE
epochs: 50
- name: ResponseSelector
```

```
epochs: 100
constrain_similarities: true
retrieval_intent: faq
```

کل config به شکل زیر می‌باشد.

```
recipe: default.v1

assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 50
  constrain_similarities: true
#- name: EntitySynonymMapper

- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۷ عملکرد مدل **LaBSE** با ۵۰ **epoch** و رویکرد ۲

همچنین برای responceها مدل دقت‌های زیر را دارا است:


```
"accuracy": 0.936,
"macro avg": {
  "precision": 0.9463333333333334,
  "recall": 0.9359999999999999,
  "f1-score": 0.9365703185703186,
  "support": 250
},
"weighted avg": {
  "precision": 0.9463333333333334,
  "recall": 0.936,
  "f1-score": 0.9365703185703186,
  "support": 250
```

```
  },
  "micro avg": {
    "precision": 0.936,
    "recall": 0.936,
    "f1-score": 0.936,
    "support": 250
  }
```

و نتایج مدل برای intentها به شکل زیر است:

```
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  }
```

مدل LaBSE با epoch=100:

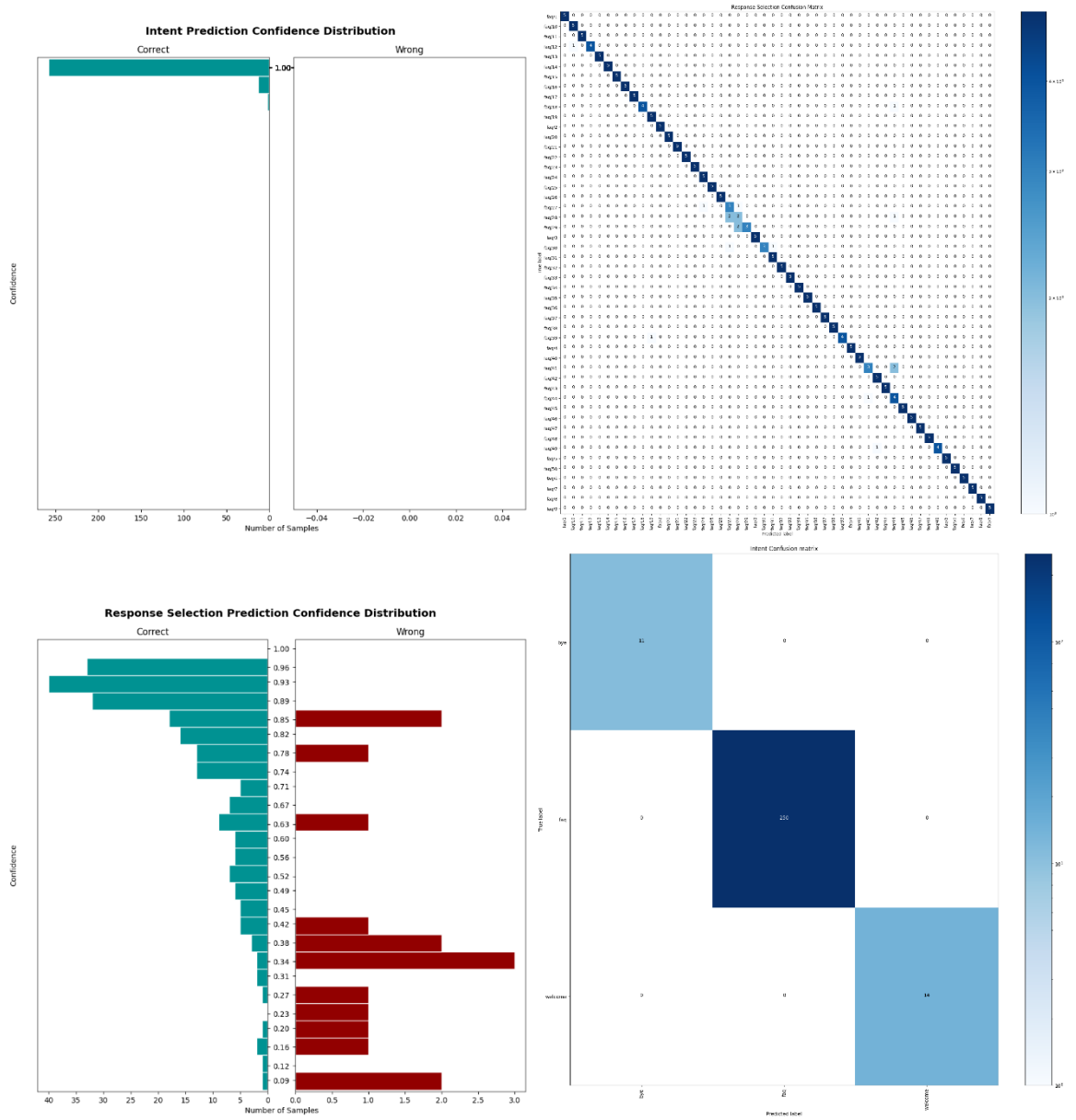
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: rasa/LaBSE
epochs: 100
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

کل config به شکل زیر می‌باشد.

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 100
  constrain_similarities: true
#- name: EntitySynonymMapper

- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۸ عملکرد مدل **LaBSE** با ۱۰۰ **epoch** و رویکرد ۲

همچنین برای responceها مدل دقت‌های زیر را دارا است:


```
"accuracy": 0.936,
"macro avg": {
  "precision": 0.9463333333333334,
  "recall": 0.936,
  "f1-score": 0.9365703185703184,
  "support": 250
},
"weighted avg": {
  "precision": 0.9463333333333334,
  "recall": 0.936,
  "f1-score": 0.9365703185703186,
  "support": 250
```

```
  },
  "micro avg": {
    "precision": 0.936,
    "recall": 0.936,
    "f1-score": 0.936,
    "support": 250
  }
```


و نتایج مدل برای intentها به شکل زیر است:


```
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  }
```


مدل LaBSE با epoch=200:

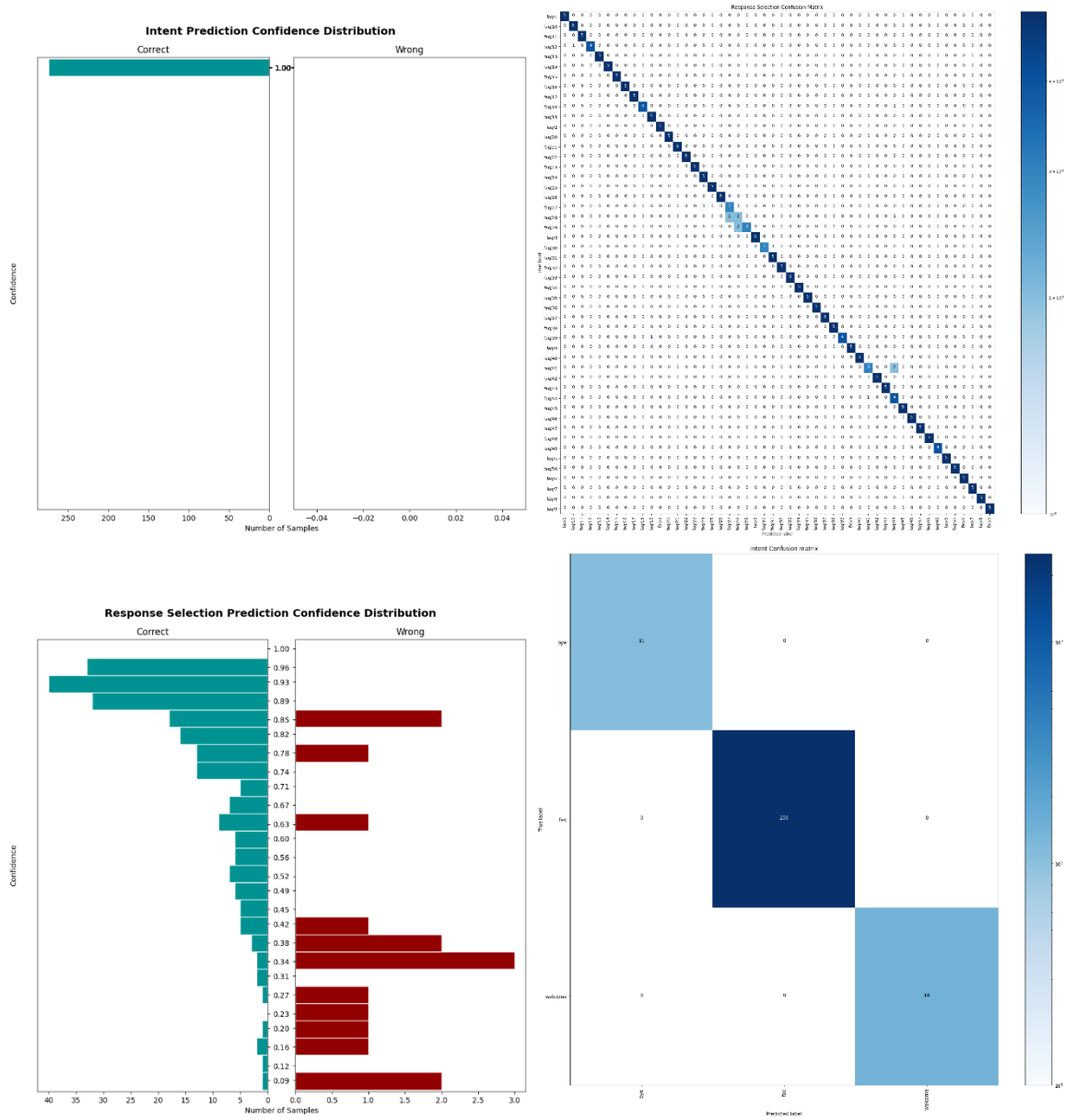
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: rasa/LaBSE
epochs: 200
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

کل config به شکل زیر می‌باشد.

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 200
  constrain_similarities: true
#- name: EntitySynonymMapper

- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۹ عملکرد مدل **LaBSE** با ۲۰۰ **epoch** و رویکرد ۲

همچنین برای responceها مدل دقت‌های زیر را دارا است:


```
"accuracy": 0.936,
"macro avg": {
  "precision": 0.9463333333333332,
  "recall": 0.9359999999999999,
  "f1-score": 0.9365703185703186,
  "support": 250
},
"weighted avg": {
  "precision": 0.9463333333333334,
  "recall": 0.936,
  "f1-score": 0.9365703185703186,
  "support": 250
```

```
  },
  "micro avg": {
    "precision": 0.936,
    "recall": 0.936,
    "f1-score": 0.936,
    "support": 250
  }
```

و نتایج مدل برای intentها به شکل زیر است:

```
   "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  }
```

مدل ParsBert با epoch=50:

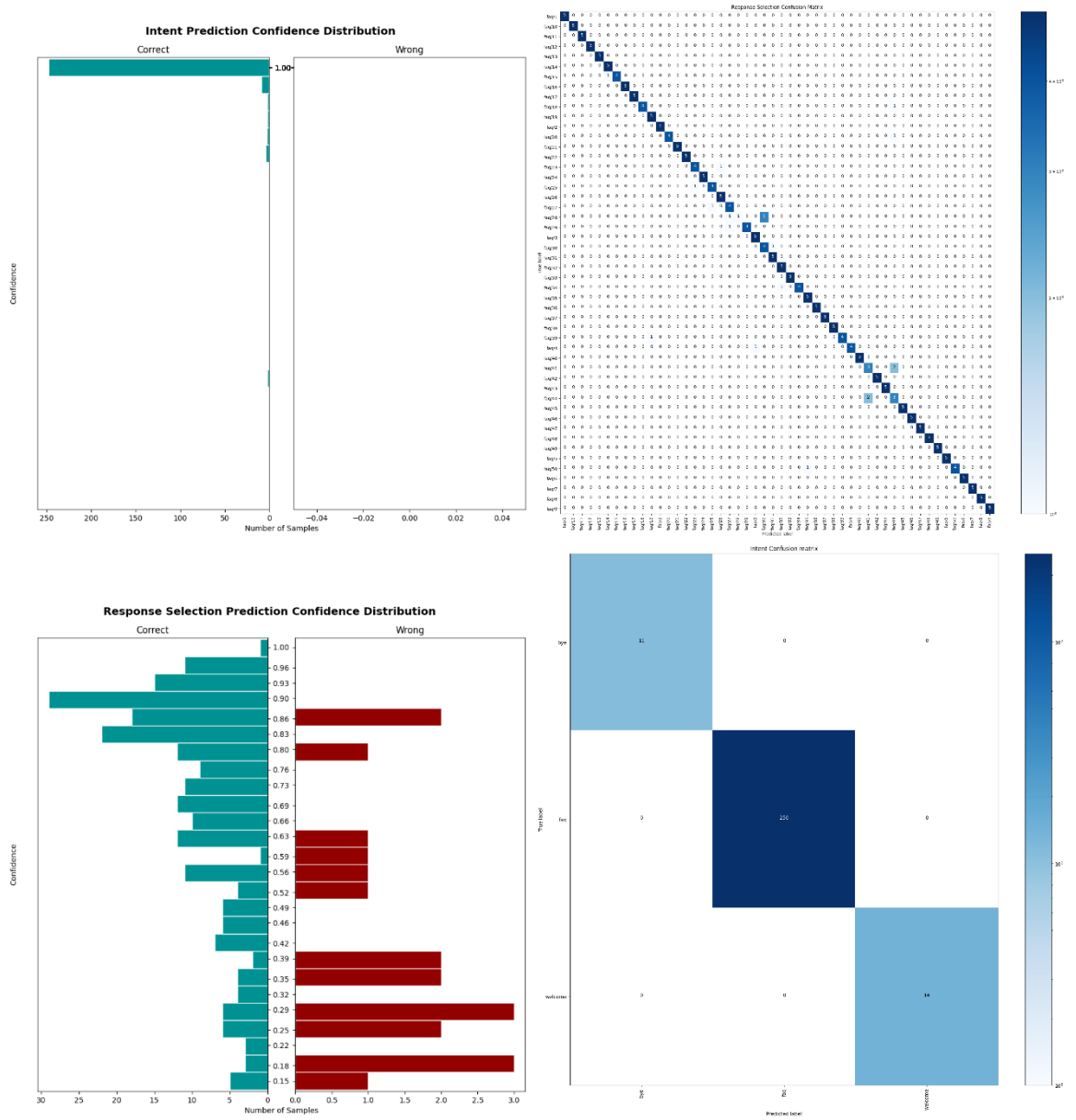
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: HooshvareLab/bert-base-parsbert-uncased
epochs: 50
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

کل config به شکل زیر می‌باشد.

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  #model_weights: rasa/LaBSE
  model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 50
  constrain_similarities: true
#- name: EntitySynonymMapper

- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۱۰ عملکرد مدل **ParsBert** با ۵۰ **epoch** و رویکرد ۲

همچنین برای responceها مدل دقت‌های زیر را دارا است:

```
"accuracy": 0.92,
"macro avg": {
  "precision": 0.9339999999999999,
  "recall": 0.92,
  "f1-score": 0.9180404040404039,
  "support": 250
},
"weighted avg": {
  "precision": 0.934,
  "recall": 0.92,
  "f1-score": 0.918040404040404,
  "support": 250
```

```
  },
  "micro avg": {
    "precision": 0.92,
    "recall": 0.92,
    "f1-score": 0.92,
    "support": 250
  }
```

و نتایج مدل برای intentها به شکل زیر است:

```
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  }
```

مدل ParsBert با epoch=100:

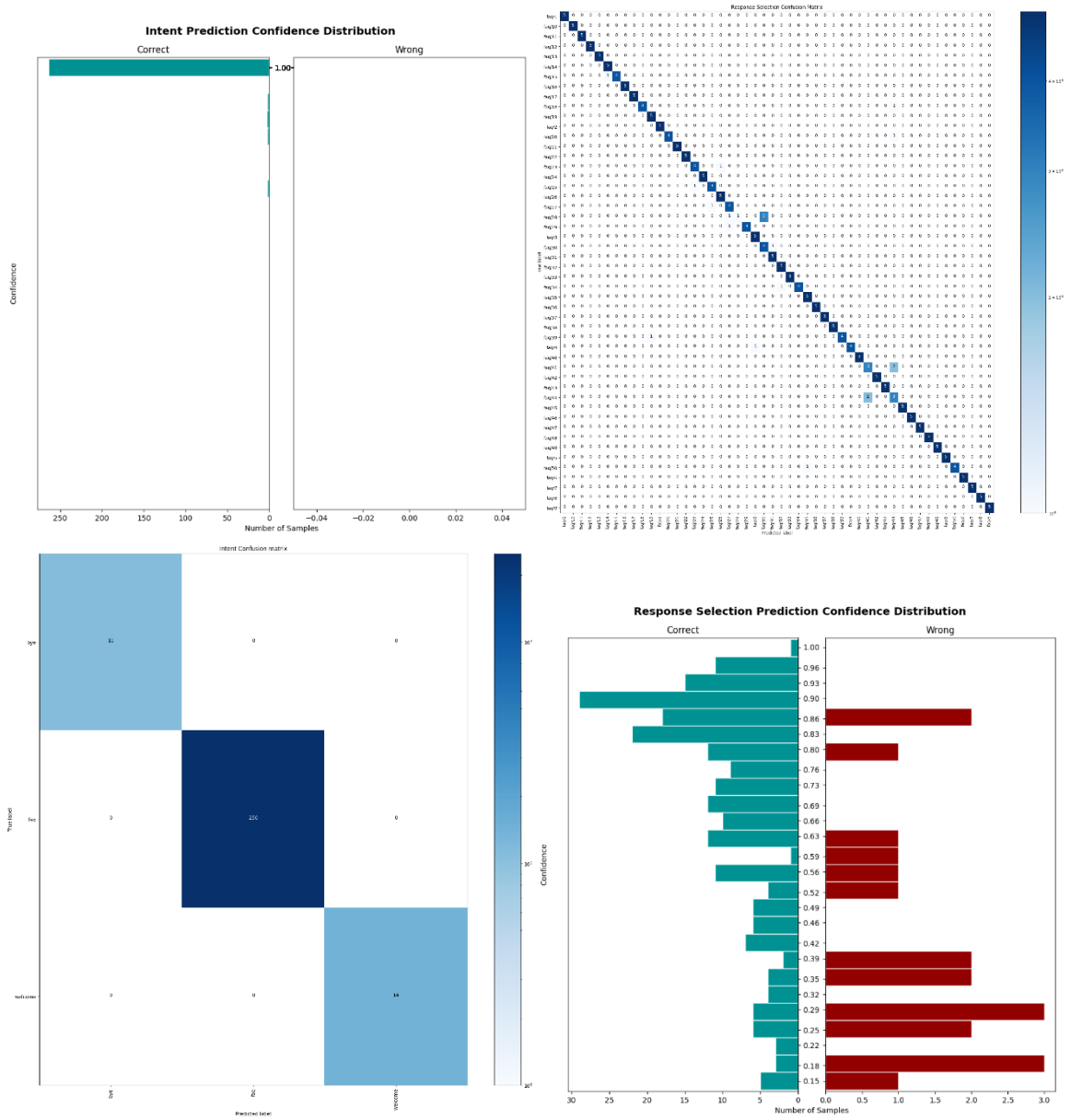
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: HooshvareLab/bert-base-parsbert-uncased
epochs: 100
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

کل config به شکل زیر می‌باشد.

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  #model_weights: rasa/LaBSE
  model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 100
  constrain_similarities: true
#- name: EntitySynonymMapper

- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۱۱ عملکرد مدل **ParsBert** با ۱۰۰ **epoch** و رویکرد ۲

همچنین برای responceها مدل دقت‌های زیر را دارا است:

```
"accuracy": 0.92,
"macro avg": {
  "precision": 0.9339999999999999,
  "recall": 0.92,
  "f1-score": 0.9180404040404041,
  "support": 250
},
"weighted avg": {
  "precision": 0.934,
  "recall": 0.92,
```

```
    "f1-score": 0.9180404040404039,
    "support": 250
  },
  "micro avg": {
    "precision": 0.92,
    "recall": 0.92,
    "f1-score": 0.92,
    "support": 250
  }
```


و نتایج مدل برای intentها به شکل زیر است:


```
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  }
```


مدل ParsBert با epoch=200:

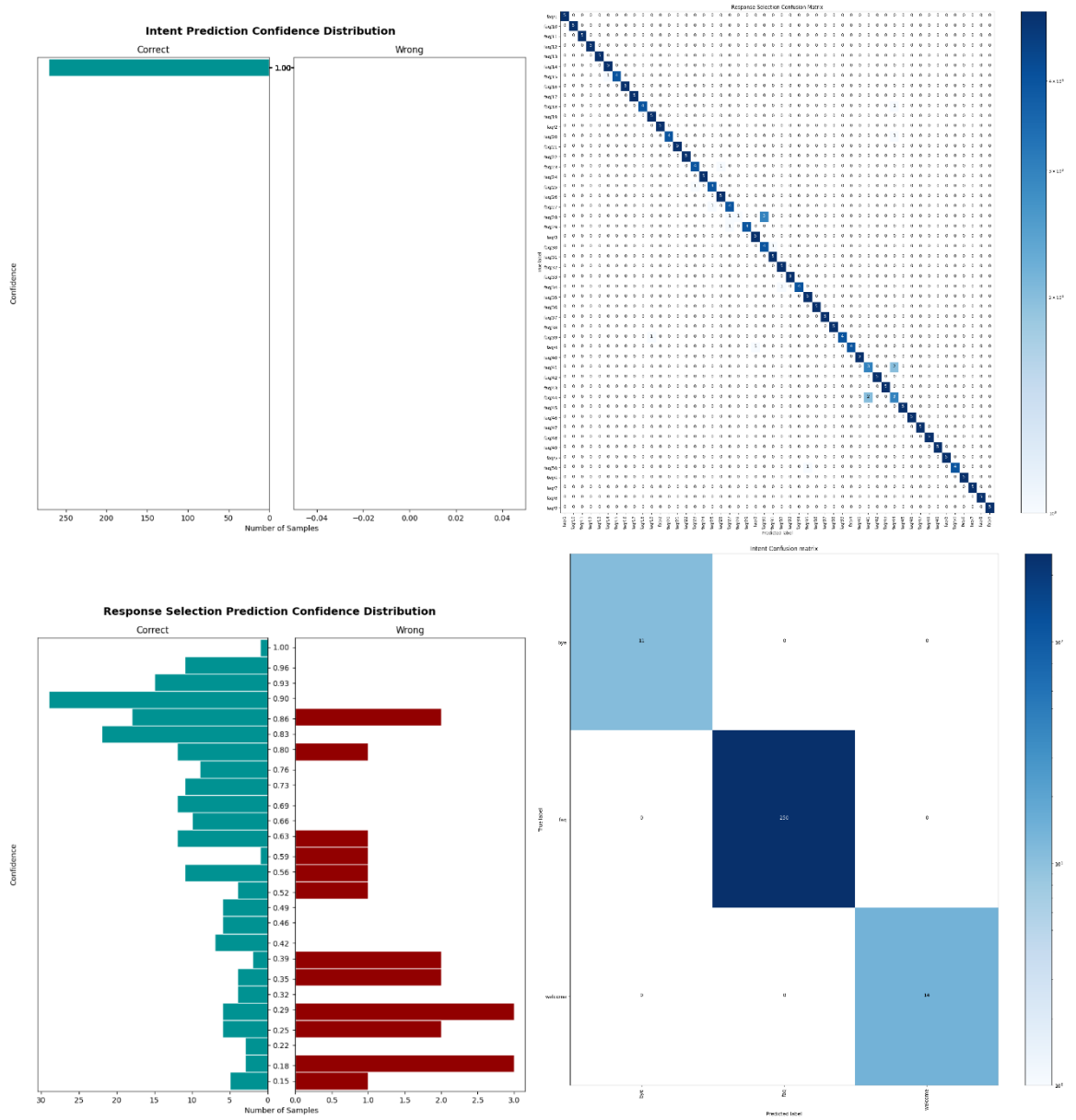
این مدل را با تغییر دو config زیر آموزش می‌دهیم:

```
model_weights: HooshvareLab/bert-base-parsbert-uncased
epochs: 200
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

کل config به شکل زیر می‌باشد.

```yaml
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel
language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  #model_weights: rasa/LaBSE
  model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 200
  constrain_similarities: true
#- name: EntitySynonymMapper

- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
  retrieval_intent: faq
```

نتایج آموزش با این مدل به صورت زیر خواهد بود:



شکل ۱۲ عملکرد مدل **ParsBert** با ۲۰۰ **epoch** و رویکرد ۲

همچنین برای responceها مدل دقت‌های زیر را دارا است:

```
"accuracy": 0.92,
"macro avg": {
  "precision": 0.934,
  "recall": 0.92,
  "f1-score": 0.9180404040404041,
  "support": 250
},
"weighted avg": {
  "precision": 0.934,
  "recall": 0.92,
  "f1-score": 0.918040404040404,
  "support": 250
```

```
  },
  "micro avg": {
    "precision": 0.92,
    "recall": 0.92,
    "f1-score": 0.92,
    "support": 250
  }
```


و نتایج مدل برای intentها به شکل زیر است:


```
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 275
  }
```


ابزار گفت‌وگوی تحت وب

از آنجا که بیشتر این ربات‌ها شبیه یکدیگر عمل می‌کردند، فقط چت با دوتا از ربات‌هایی که با ۱۰۰ دوره آموزش دیده‌اند یک بار برای LaBSe و یک بار با ParsBert برای هر رویکرد قرار داده شده است.

رویکرد ۱:

شکل ۱۳ چت با ربات و رویکرد ۱ **ParsBert**

شکل ۱۴ چت با ربات و رویکرد ۱ **LaBSE**

رویکرد ۲:

شکل ۱۵ چت با ربات و رویکرد ۲ **ParsBert**

شکل ۱۶ چت با ربات و رویکرد ۲ **LaBSE**

پاسخ ۲ – استخراج مقادیر ارزش‌ها

سه رقم آخر شماره دانشجویی من ۱۷۳ می‌باشد، پس داریم:

$$173\%6 + 1 = 6$$

پس سناریو شماره‌ی ۶ را اینجا انجام خواهیم داد که مربوط به سناریوی استخراج مقادیر ارزش های نام شهر مقصد و تاریخ پرواز می‌باشد.

آماده‌سازی دادگان و آموزش مدل

در این بخش از آنجا که باید داده‌ها را خودمان درست کنیم به صورت زیر عمل کردیم:

ابتدا intentهای مربوط به هرکار را تعریف می‌کنیم که شامل موارد زیر می‌باشد:

- Welcome: مربوط به پیام ابتدایی کاربر است که درخواست اولیه خود را بیان می‌کند.
- Get_details: در اینجا مقصد و زمان سفر توسط کاربر بیان می‌شود و با استفاده از این پیام slotهای مربوط پر می‌شود.
- Confirm: کاربر تایید می‌کند که ربات منظور او را درست فهمیده است.
- Deny: کاربر مشخص می‌کند که ربات منظور او را درست نفهمیده است.

این intentها در فایل nlu.yml قرار داده شده است. همچنین تعدادی example نیز قرار داده شده است که slotهای date و city باید پر شوند که این مقادیر در [] قرار دارند.

در domain.yml نیز intentها و slotهایی که باید پر شوند تعریف شده است.

دو slot برای این وظیفه داریم:

- تاریخ: برای اینکه تاریخ ساختار مشخصی دارد و کاربران تاریخ‌های مختلفی می‌توانند بدهند از regex به فرمت زیر استفاده کرده‌ایم که در فایل nlu.yml قرار دارد.

```
- regex: date
  examples: |
    - ^\d{4}\/(1[0-2]|0?[1-9])\/(3[01]|[12]\d|0?[1-9])$
```

- شهر: که تعدادی شهر به عنوان نمونه در exampleها قرار داده شده است و بقیه با استفاده از CRF شناسایی می‌شوند.

همچنین برای utteranceهای مختلف پاسخ‌های ربات در این فایل قرار داده شده است. که با توجه به اسلات‌های پر شده این پاسخ‌ها تغییر می‌کنند و با توجه به هر intent یک پاسخ به صورت تصادفی انتخاب می‌شود.

```
responses:
```

```
  utter_welcome:
  - text: "سلام، چه کمکی از دستم برمیاد؟"
  - text: "سلام، بفرمایید؟"
  - text: "سلام، لطفا مقصد و تاریخ سفر خود را اعلام کنید"
  utter_get_details:
  - text: "آیا به دنبال بلیت هواپیما به مقصد {city} در تاریخ {date} هستید؟"
  - text: "میخوای در تاریخ {date} به شهر {city} سفر کنی؟"

  utter_confirm:
  - text: "در حال جست و جوی بلیت برای شما"
  - text: "بله به دنبال بلیت مورد نظر برای شما هستیم"

  utter_deny:
  - text: "لطفا یک بار دیگر اطلاعات مورد نظر را وارد کنید"
  - text: "لطفا دوباره مقصد و تاریخ خود را اعلام کنید"
```

در فایل rule.yml نیز مشخص شده است که هر intent کدام پاسخ را توسط ربات داشته باشد. برای مثال intent مربوط به welcome با utterance مربوط به utter_welcome پاسخ می‌گیرد.

```
rules:
- rule: welcome user
  steps:
  - intent: welcome
  - action: utter_welcome
- rule: get details
  steps:
  - intent: get_details
  - action: utter_get_details
- rule: confirm
  steps:
  - intent: confirm
  - action: utter_confirm
- rule: deny
  steps:
  - intent: deny
  - action: utter_deny
```

علاوه بر فایل‌های بالا در این بخش stories.yml نیز اضافه می‌شود که در آن مسیرهای مختلفی برای ربات می‌توانیم تعیین کنیم در اینجا دو مسیر که یکی ربات می‌تواند بلیتی برای شخص تهیه کنید و یک مسیر که خرید با شکست مواجه می‌شود تعریف کرده‌ایم.

```
stories:

- story: confirm path
  steps:
  - intent: welcome
  - action: utter_welcome
```

```
  - intent: get_details
  - action: utter_get_details
  - intent: confirm
  - action: utter_confirm

- story: deny path
  steps:
  - intent: welcome
  - action: utter_welcome
  - intent: get_details
  - action: utter_get_details
  - intent: deny
  - action: utter_deny
```

همچنین فایل cofig.yml نیز به این صورت تغییر می‌کند که CRFEntityExtractor اضافه شده است که در سوال یک توضیح داده شد.

```
# The config recipe.
# https://rasa.com/docs/rasa/model-configuration/
recipe: default.v1
assistant_id: 20230627-201605-chamfered-roundel

language: fa

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4

- name: LanguageModelFeaturizer
  model_name: bert
  model_weights: rasa/LaBSE
  #model_weights: HooshvareLab/bert-base-parsbert-uncased

- name: DIETClassifier #indent classifer
  epochs: 100
  constrain_similarities: true
#- name: EntitySynonymMapper
- name: RegexEntityExtractor
- name: CRFEntityExtractor
```

شکل ۱۷ فایل **config** برای فروش بلیت

پیاده‌سازی و تحلیل نتایج

پاسخ سوال‌ها:

- در فایل config تغییرات زیر انجام شده است:
  - از مدل LaBSe استفاده کردیم، چون در سوال اول بهتر جواب می‌داد.
  - برای ۱۰۰ تا epoch آموزش را انجام دادیم.

- CRFEntityExtractor به pipeline اضافه شده است، از آنجا که در جملات قصد داریم entityها را پیدا کنیم و slotهای لازم را پر کنیم مشخص است که دلیل استفاده از این بخش چیست.
- RegexEntityExtractor نیز با استفاده از عبارات منظم سعی می‌کند که entity از پاسخ کاربر استخراج کند.

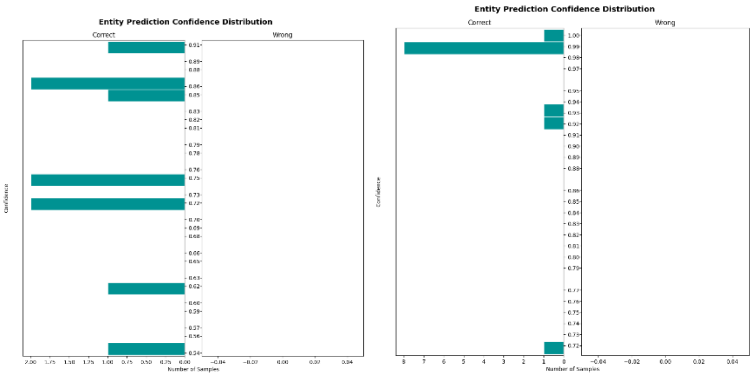
- برای سناریو ۶، ما دو entity شهر و تاریخ داریم که به ترتیب توسط crf و regex استخراج می‌شوند. مشخص است که crf با توجه به متن و context قابلیت استخراج entity دارد اما regex باید الگوی از پیش تعریف شده داشته باشد که برای تاریخ مناسب است.
- نتایج گفت‌وگو به شکل زیر می‌باشد:

مشخص است که ربات توانسته slotها را پر کند و درخواست کاربر برای سفر را تشخیص داده است.

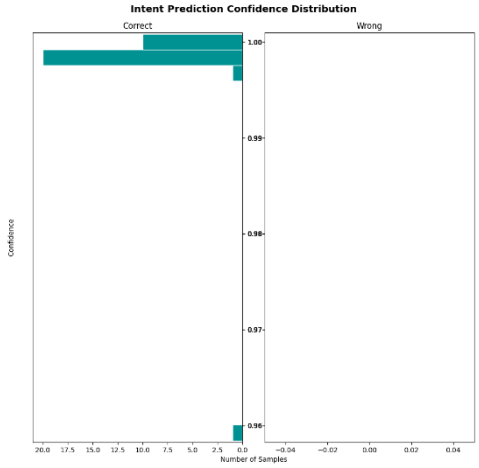
- حال با فرمان زیر عملیات k-fold را انجام می‌دهیم.

```
rasa test  --cross-validation --folds 3
```

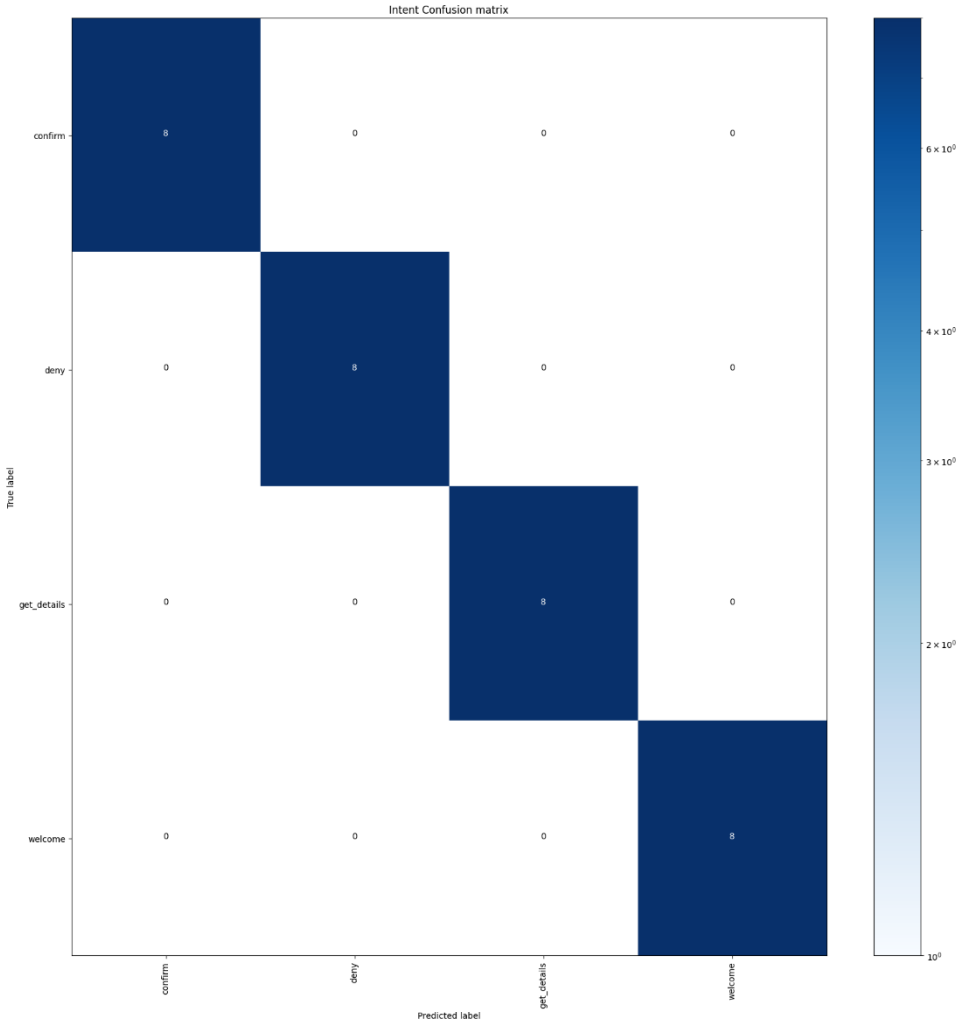
نتایج مدل به شکل زیر می‌باشد:



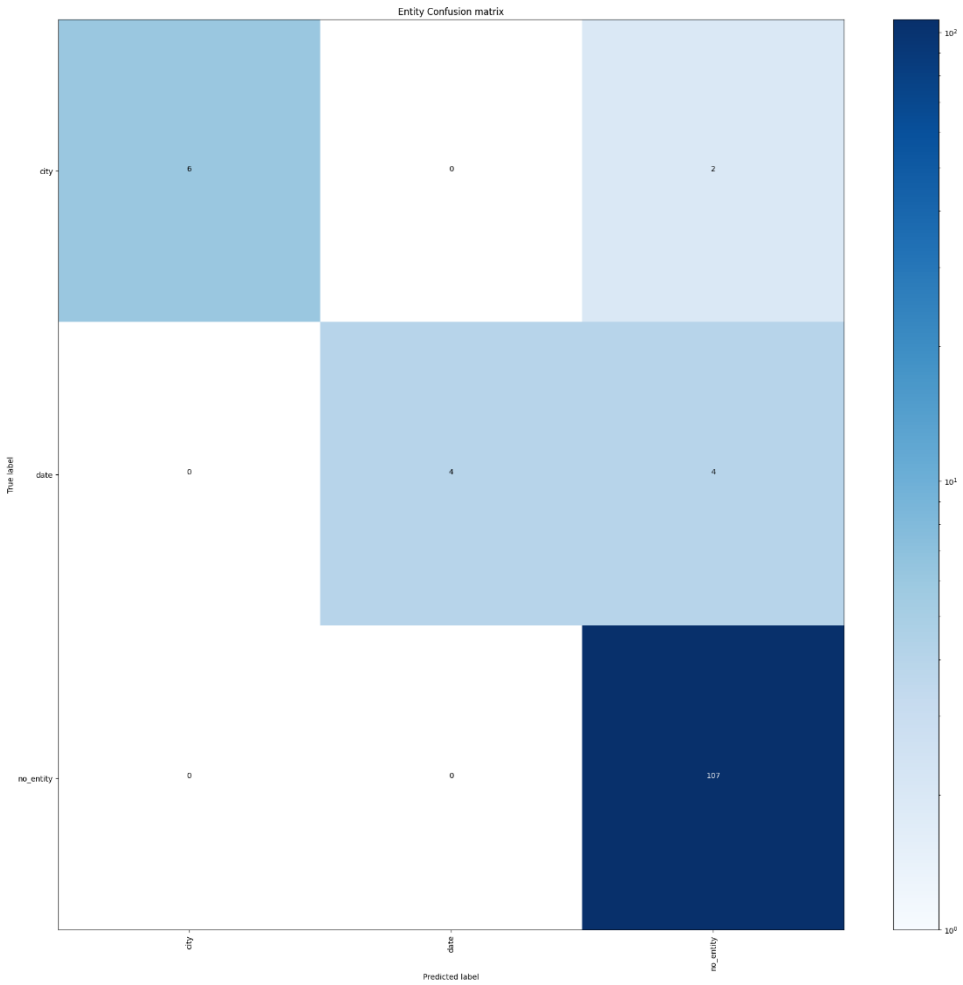
شکل ۱۸ **entity predication confidence**



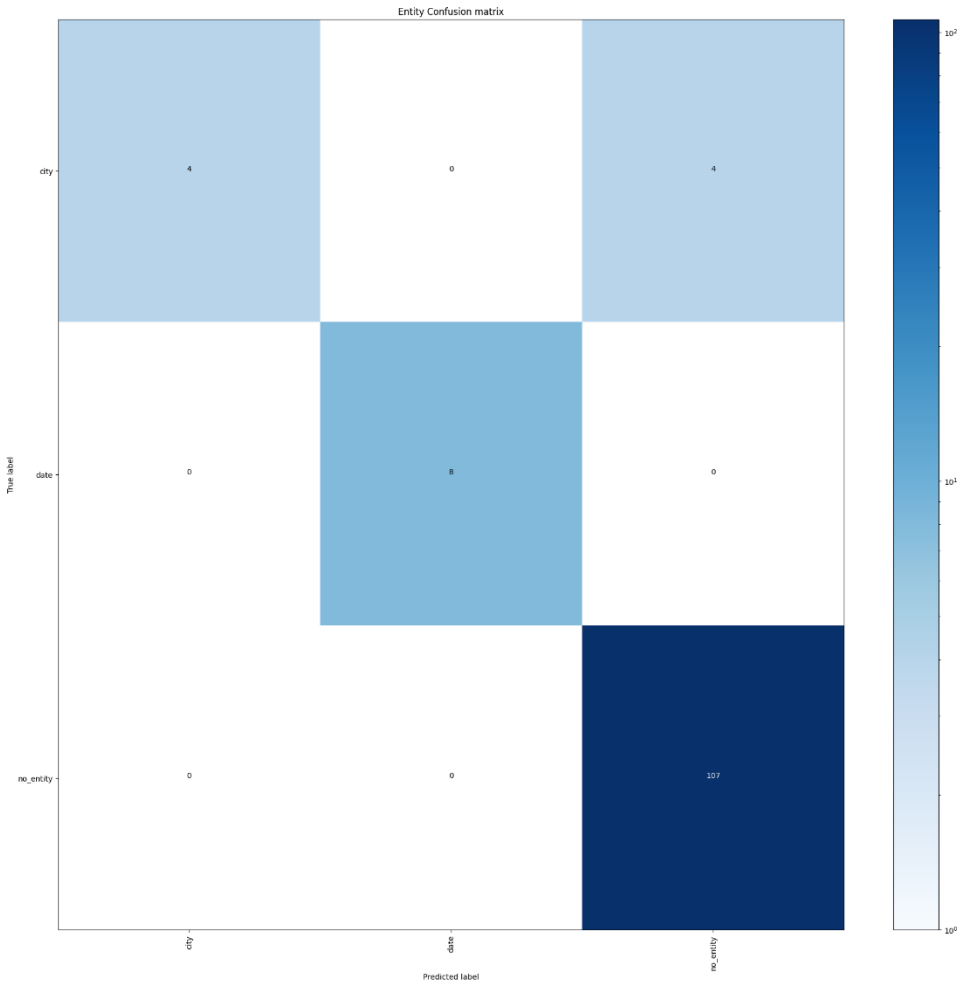
شکل ۱۹ **Intent predication confidence**



Intent confusion matrix for ticket شکل ۲۰



شکل ۲۱ **entity confusion matrix for ticket**



شکل ۲۲ **entity confusion matrix for ticket**

همچنین دقت مدل CRF به شرح زیر است:


```
{
  "date": {
    "precision": 1.0,
    "recall": 0.5,
    "f1-score": 0.6666666666666666,
    "support": 8,
    "confused_with": {}
  },
  "city": {
    "precision": 1.0,
    "recall": 0.75,
    "f1-score": 0.8571428571428571,
    "support": 8,
    "confused_with": {}
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 0.625,
```

```
    "f1-score": 0.7692307692307693,
    "support": 16
  },
  "macro avg": {
    "precision": 1.0,
    "recall": 0.625,
    "f1-score": 0.7619047619047619,
    "support": 16
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 0.625,
    "f1-score": 0.7619047619047619,
    "support": 16
  },
  "accuracy": 0.9512195121951219
}
```


دقت مدل dietclassifer نیز به شرح زیر است:


```
{
  "date": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "city": {
    "precision": 1.0,
    "recall": 0.5,
    "f1-score": 0.6666666666666666,
    "support": 8,
    "confused_with": {}
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 0.75,
    "f1-score": 0.8571428571428571,
    "support": 16
  },
  "macro avg": {
    "precision": 1.0,
    "recall": 0.75,
    "f1-score": 0.8333333333333333,
    "support": 16
  },
  "weighted avg": {
    "precision": 1.0,
```

```
    "recall": 0.75,
    "f1-score": 0.8333333333333333,
    "support": 16
  },
  "accuracy": 0.967479674796748
}
```


و برای intentها نیز داریم:


```
{
  "get_details": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "confirm": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "deny": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "welcome": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 8,
    "confused_with": {}
  },
  "accuracy": 1.0,
  "macro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 32
  },
  "weighted avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
```

```
    "support": 32
  },
  "micro avg": {
    "precision": 1.0,
    "recall": 1.0,
    "f1-score": 1.0,
    "support": 32
  }
}
```

- مقایسه‌ی RegexEntityExtractor و CRFEntityExtractor:

هر کدام از استخراج کننده‌های Regex و CRF در Rasa به روش خود برای استخراج مقادیر ارزش‌ها از متن ورودی پرداخت می‌کنند. بیایید هر دو را مقایسه کنیم:

RegexEntityExtractor:

- عملکرد: RegexEntityExtractor بر مبنای الگوهای قابل تعریفی که ما تعیین می‌کنیم، مقادیر entityها را استخراج می‌کند. یعنی با استفاده از الگوهای عبارت منظمی که تعریف شده است. می‌تواند راه حل سریع و ساده‌ای برای استخراج ارزش‌ها از متن ورودی باشد.

CRFEntityExtractor:

- عملکرد: CRFEntityExtractor از روش‌های یادگیری ماشینی برای استخراج entityها استفاده می‌کند. با استفاده از مدل CRF، این روش، مدلی را بر اساس داده‌های آموزش می‌دهد که برچسب‌های entityها را برای متن ورودی مشخص می‌کند. سپس مدل آموزش دیده شده از الگوها و زمینه‌ی[1] متن استفاده کرده و برچسب‌های entityها را برای متن ورودی تخمین می‌زند.

مزایا و محدودیت‌ها:

- RegexEntityExtractor:

  - مزایا:

    - ساده و سریع در تعریف الگوهای استخراج ارزش‌ها.

    - مناسب برای الگوهای ثابت و قابل پیش‌بینی.

  - محدودیت‌ها:

    - نیاز به تعیین دقیق الگوها و استفاده از عبارات منظم.

[1] context

- قابلیت استفاده محدودتر در مواقعی که الگوها پیچیده یا تغییرپذیر هستند.

- CRFEntityExtractor:

- مزایا:

- قابلیت استفاده در الگوهای پیچیده و پویا.

- توانایی استفاده از زمینه‌ی متن برای استخراج ارزش‌ها.

- محدودیت‌ها:

- نیاز به داده‌های آموزشی برچسب‌گذاری شده برای مدل CRF.

- پیچیدگی بیشتر در آموزش و بهینه‌سازی مدل.

بنابراین، برای استخراج مقادیر ارزش‌ها، RegexEntityExtractor مناسب است زمانی که الگوهای ثابت و قابل پیش‌بینی وجود دارد. اگر الگوها پیچیده‌تر هستند و یا قابلیت تغییر پذیری بیشتری دارند، CRFEntityExtractor می‌تواند یک راه حل مناسب‌تر باشد. در اینجا شهرها وابسته به متن هستند و الگوی خاصی ندارند لذا بهتر است از CRF استفاده کنیم، اما برای تاریخ که یک الگوی ثابت دارد بهتر است از Regex استفاده شود.