

# Processamento de Linguagem Natural

Projeto Final - Chatbot com Rasa

Gabriel de Melo Evangelista

Lucas dos Reis Silva

Maria Luísa Leandro de Lima

# Classificador e Extrator

Resultados

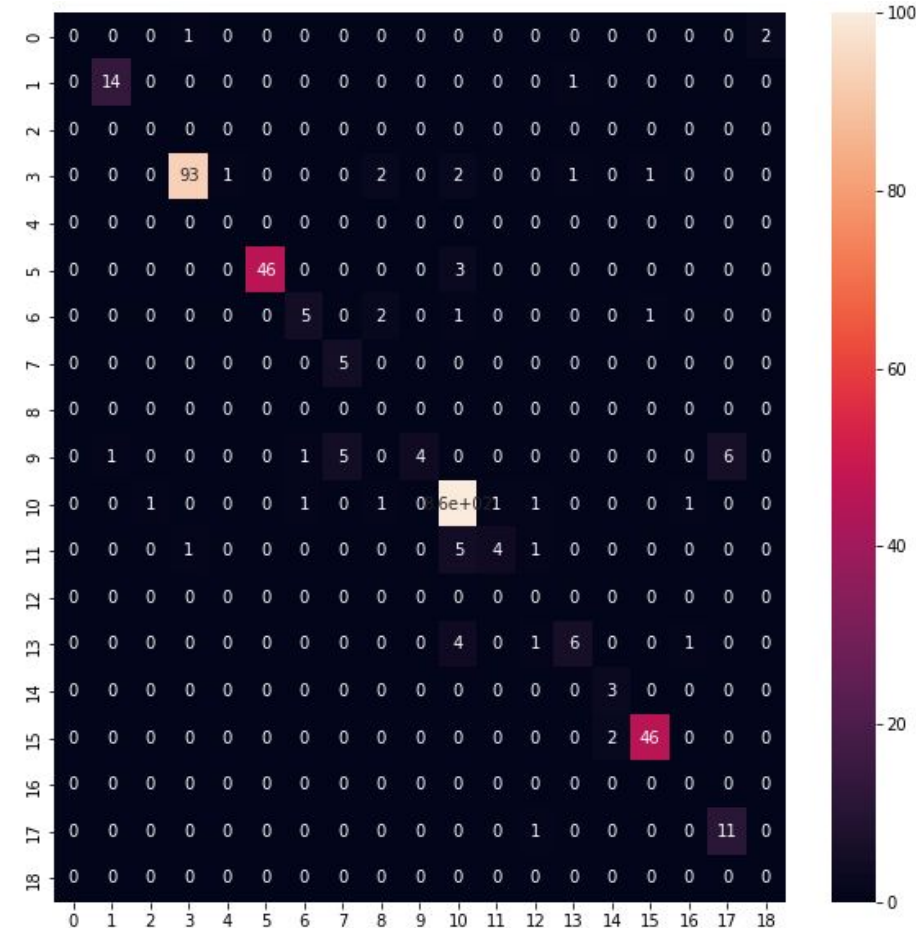
# Classificador

```
def create_model(type, embedding_dims, embedding_size, input_length, units, dropout, labels):  
    model = Sequential()  
    model.add(Embedding(embedding_dims, embedding_size, input_length=input_length))  
    model.add(type(units, return_sequences=True)) #p/ mais de uma camada  
    model.add(Dropout(dropout))  
    model.add(type(units, return_sequences=False))  
    model.add(Dropout(dropout))  
    model.add(Dense(labels, activation='softmax'))  
    return model
```

```
INPUT_LENGTH = 48  
EMBEDDING_SIZE = 100  
UNITS = 128  
DROPOUT = 0.2  
### Training parameters  
LOSS = "categorical_crossentropy"  
OPTIMIZER = "adam"  
BATCH_SIZE = 32  
EPOCHS = 8  
LABELS = 26  
EMBEDDING_DIMS = len(dict['token_ids'])+1
```

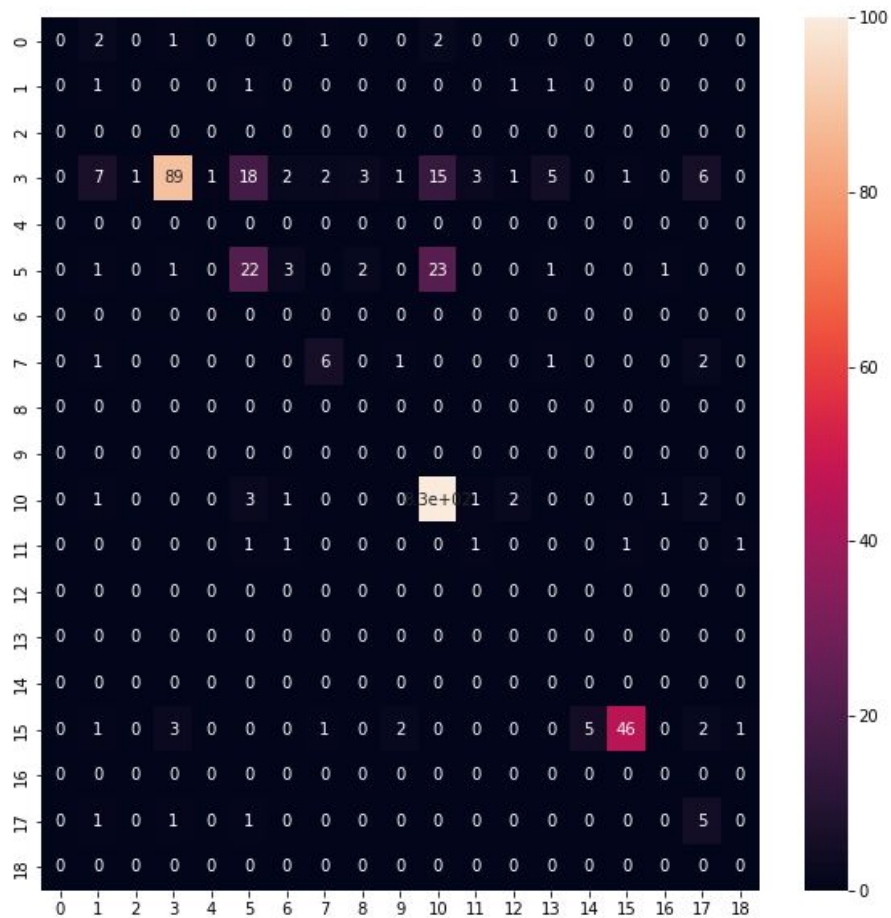
```
Done loading: atis.train.pkl  
    samples: 4978  
    vocab_size: 943  
    slot count: 129  
    intent count: 26  
Done loading: atis.test.pkl  
    samples: 893  
    vocab_size: 943  
    slot count: 129  
    intent count: 26
```

# LSTM

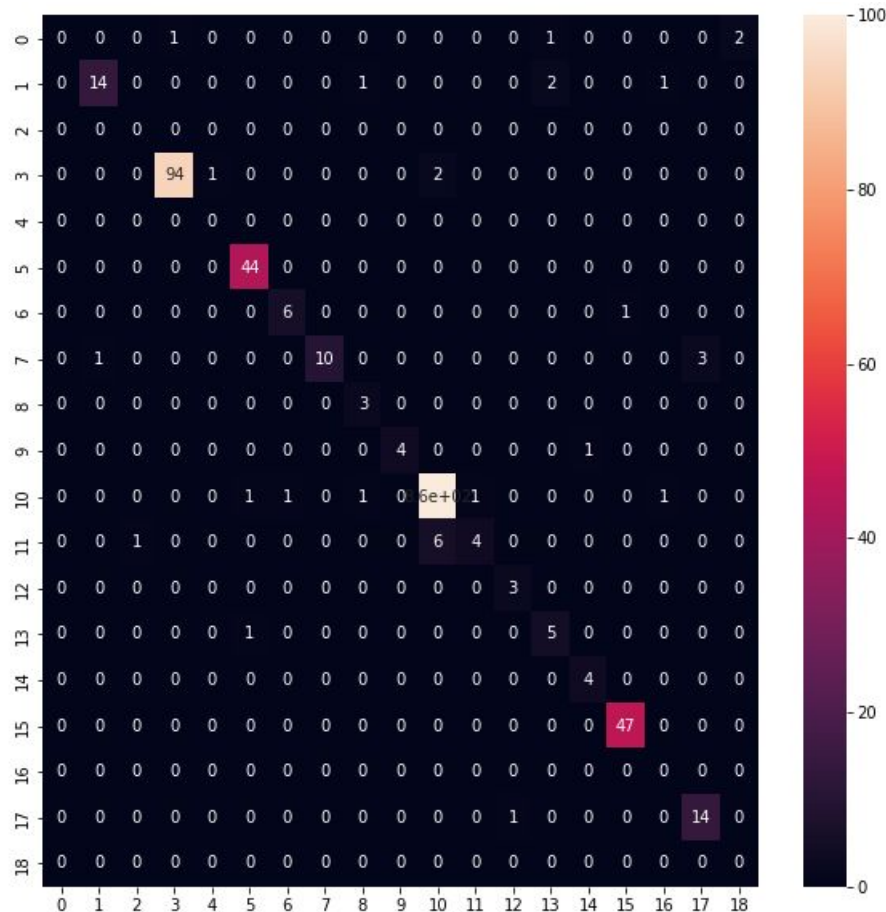


	precision	recall	f1-score	support
0	0.00	0.00	0.00	3
1	0.93	0.93	0.93	15
2	0.00	0.00	0.00	0
3	0.98	0.93	0.95	100
4	0.00	0.00	0.00	0
6	1.00	0.94	0.97	49
8	0.71	0.56	0.63	9
9	0.50	1.00	0.67	5
11	0.00	0.00	0.00	0
13	1.00	0.24	0.38	17
14	0.98	0.99	0.99	863
15	0.80	0.36	0.50	11
17	0.00	0.00	0.00	0
19	0.75	0.50	0.60	12
20	0.60	1.00	0.75	3
21	0.96	0.96	0.96	48
23	0.00	0.00	0.00	0
24	0.65	0.92	0.76	12
25	0.00	0.00	0.00	0
accuracy			0.95	1147
macro avg	0.52	0.49	0.48	1147
weighted avg	0.97	0.95	0.95	1147

# RNN



	precision	recall	f1-score	support
0	0.00	0.00	0.00	6
1	0.07	0.25	0.11	4
2	0.00	0.00	0.00	0
3	0.94	0.57	0.71	155
4	0.00	0.00	0.00	0
6	0.48	0.41	0.44	54
8	0.00	0.00	0.00	0
9	0.60	0.55	0.57	11
11	0.00	0.00	0.00	0
13	0.00	0.00	0.00	0
14	0.95	0.99	0.97	843
15	0.20	0.20	0.20	5
17	0.00	0.00	0.00	0
19	0.00	0.00	0.00	0
20	0.00	0.00	0.00	0
21	0.96	0.75	0.84	61
23	0.00	0.00	0.00	0
24	0.29	0.62	0.40	8
25	0.00	0.00	0.00	0
accuracy			0.87	1147
macro avg	0.24	0.23	0.22	1147
weighted avg	0.91	0.87	0.88	1147



	precision	recall	f1-score	support
0	0.00	0.00	0.00	4
1	0.93	0.78	0.85	18
2	0.00	0.00	0.00	0
3	0.99	0.97	0.98	97
4	0.00	0.00	0.00	0
6	0.96	1.00	0.98	44
8	0.86	0.86	0.86	7
9	1.00	0.71	0.83	14
11	0.60	1.00	0.75	3
13	1.00	0.80	0.89	5
14	0.99	0.99	0.99	869
15	0.80	0.36	0.50	11
17	0.75	1.00	0.86	3
19	0.62	0.83	0.71	6
20	0.80	1.00	0.89	4
21	0.98	1.00	0.99	47
23	0.00	0.00	0.00	0
24	0.82	0.93	0.87	15
25	0.00	0.00	0.00	0
accuracy			0.97	1147
macro avg	0.64	0.64	0.63	1147
weighted avg	0.98	0.97	0.97	1147

# Extrator

```
Done loading: atis.train.pkl
  samples: 4978
  vocab_size: 943
  slot count: 129
  intent count: 26
Done loading: atis.test.pkl
  samples: 893
  vocab_size: 943
  slot count: 129
  intent count: 26
```

```
BATCH_SIZE = 64
EMBEDDING_DIM = 128
UNITS = 128
EPOCHS= 4
LABEL = len(i2l)+1
VOCABULARY = len(dict['token_ids'])+1
INPUT_LENGTH = 42
INPUT_DIM = len(x_text)+1
OUTPUT_DIM=64
```

```
model = Sequential()
model.add(Embedding(input_dim=INPUT_DIM, output_dim=UNITS, input_length=INPUT_LENGTH))
model.add(Bidirectional(GRU(units=UNITS, return_sequences=True, dropout=0.3, recurrent_dropout=0.3), merge_mode = 'concat'))
model.add(GRU(units=UNITS, return_sequences=True, dropout=0.5, recurrent_dropout=0.5))
model.add(TimeDistributed(Dense(LABEL, activation="relu")))
model.summary()
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```



	precision	recall	f1-score	support	26	0.17	0.93	0.29	15
					27	0.00	0.00	0.00	0
0	0.00	0.00	0.00	22	28	0.00	0.00	0.00	0
1	0.02	1.00	0.04	1	29	0.00	0.00	0.00	0
2	0.00	0.00	0.00	0	30	0.00	0.00	0.00	0
3	0.06	1.00	0.11	1	31	0.27	0.88	0.41	49
4	0.00	0.00	0.00	0	32	0.04	1.00	0.07	1
5	0.33	0.75	0.46	8	33	0.14	1.00	0.25	12
6	0.04	0.25	0.07	4	34	0.00	0.00	0.00	0
7	0.00	0.00	0.00	0	35	0.47	0.94	0.62	507
8	0.00	0.00	0.00	0	36	0.00	0.00	0.00	0
9	0.13	0.92	0.22	12	37	0.00	0.00	0.00	0
10	0.10	1.00	0.19	18	38	0.08	1.00	0.15	12
11	0.16	0.65	0.25	17	39	0.50	0.87	0.64	136
12	0.00	0.00	0.00	0	40	0.34	0.93	0.50	28
13	0.02	1.00	0.05	1	41	0.00	0.00	0.00	0
14	0.00	0.00	0.00	0	42	0.26	1.00	0.42	15
15	0.00	0.00	0.00	0	43	0.99	0.77	0.87	10737
16	0.22	1.00	0.36	22	44	0.06	1.00	0.11	1
17	0.00	0.00	0.00	0	45	0.00	0.00	0.00	0
18	0.29	0.50	0.36	4	46	0.76	0.95	0.85	215
19	0.17	0.82	0.28	17	47	0.01	1.00	0.02	1
20	0.02	1.00	0.04	1	48	0.00	0.00	0.00	0
21	0.00	0.00	0.00	0	49	0.00	0.00	0.00	0
22	0.00	0.00	0.00	0	50	0.00	0.00	0.00	0
23	0.23	0.76	0.36	25	51	0.07	1.00	0.12	3
24	0.00	0.00	0.00	0	52	0.10	1.00	0.17	4
25	0.54	0.62	0.58	76	53	0.85	0.87	0.86	998

54	0.00	0.00	0.00	0
55	0.00	0.00	0.00	0
56	0.33	0.78	0.46	18
accuracy				
				0.79
macro avg				
				0.14
weighted avg				
				0.93
				0.48
				0.18
				0.84
				12981
				12981
				12981

A matriz de confusão é muito grande! Vamos ao [GitHub](#)

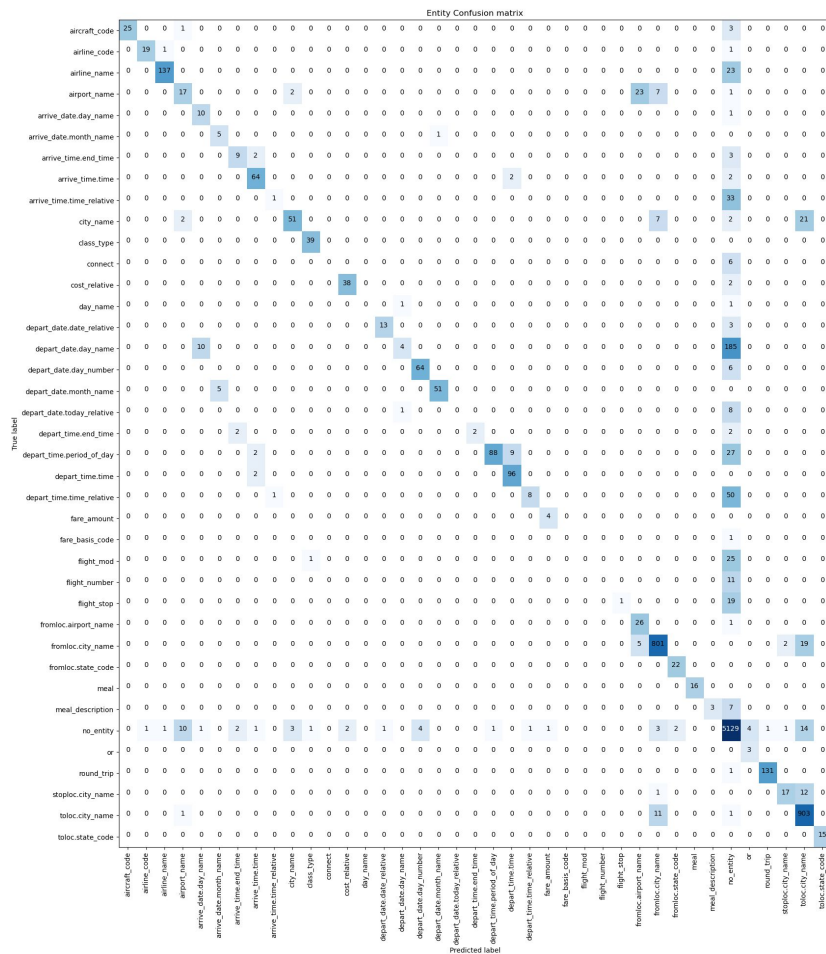


# Chatbot com o Rasa

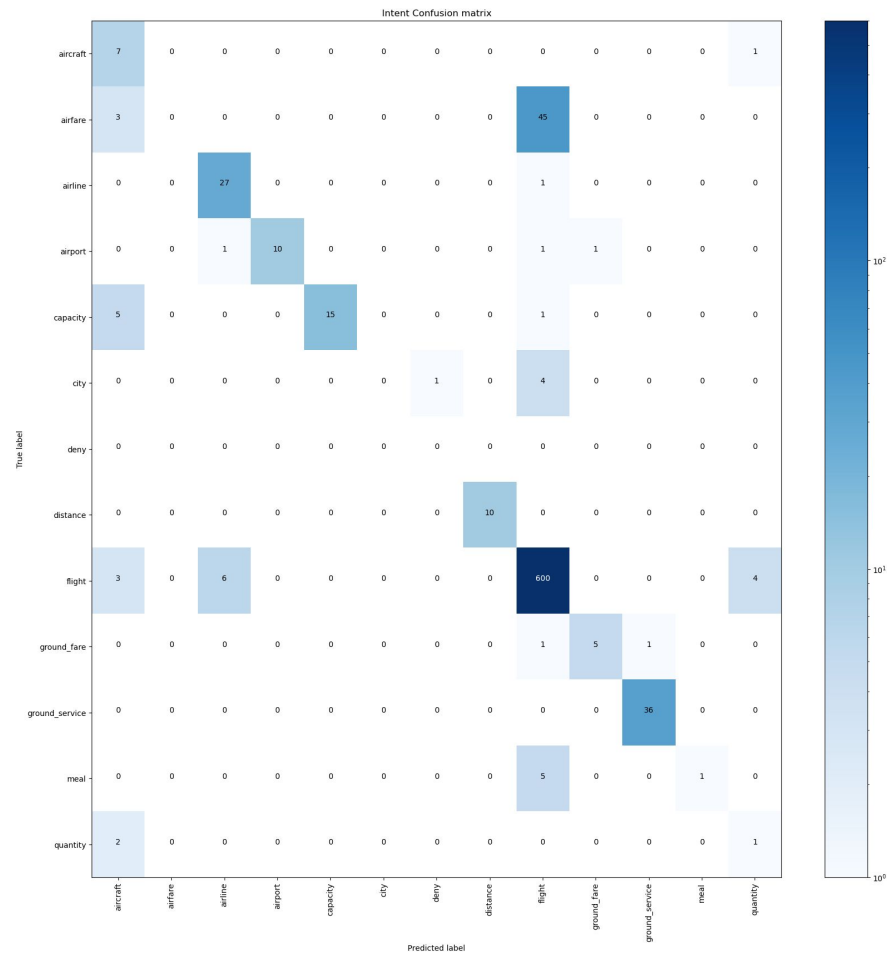
# Classificador

Colocar prints do rasa teste

```
"micro avg": {  
  "precision": 0.927731673582296,  
  "recall": 0.822501532801962,  
  "f1-score": 0.8719532011699709,  
  "support": 3262  
},  
"macro avg": {  
  "precision": 0.7111569637312417,  
  "recall": 0.618799018681978,  
  "f1-score": 0.6118535524885375,  
  "support": 3262  
},  
"weighted avg": {  
  "precision": 0.897342279375854,  
  "recall": 0.822501532801962,  
  "f1-score": 0.8266082646461117,  
  "support": 3262  
}
```



# Extrator



# Stories

Temos apenas dois stories, pois o form permite captar vários slots em uma única chamada.

```
- story: principalVoo
  steps:
  - intent: flight
  - action: flight_form
  - active_loop: flight_form
  - active_loop: null
  - action: action_api_test
  - checkpoint: check_flow_finished
```

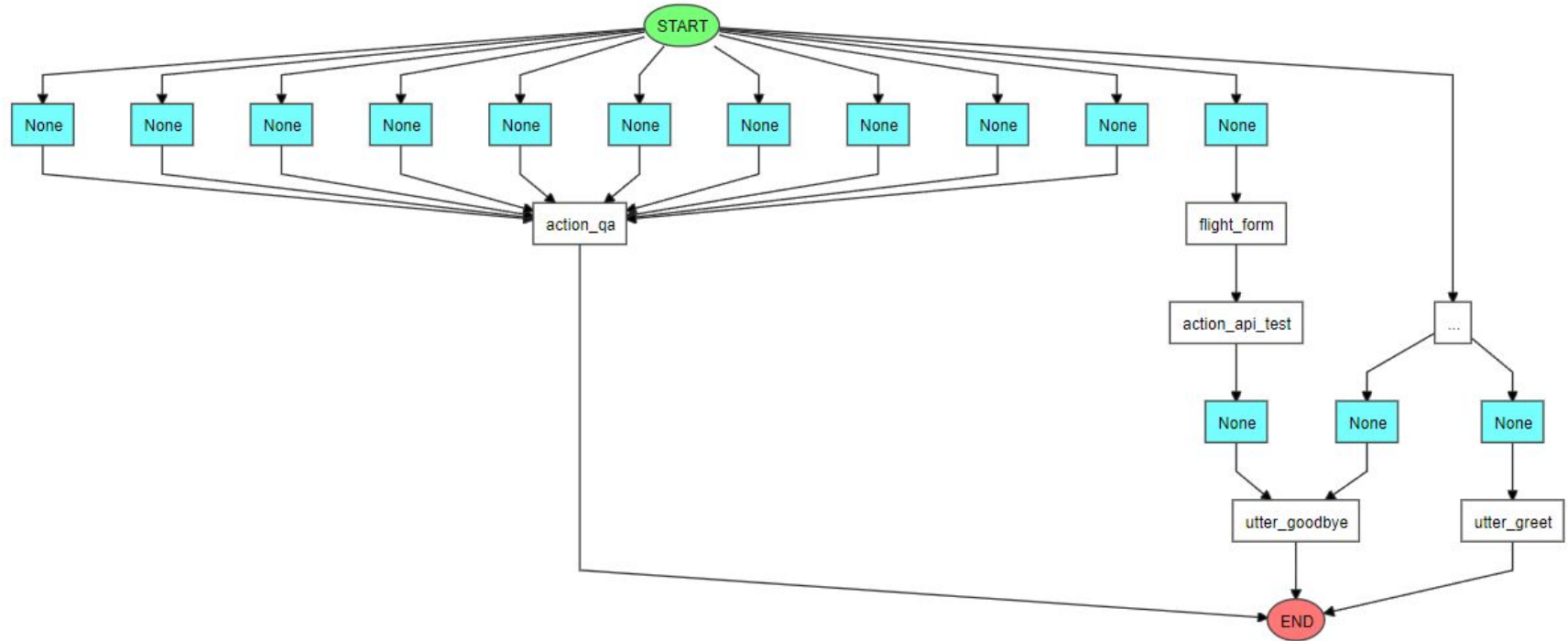
```
- story: finish flow
  steps:
  - checkpoint: check_flow_finished
  - intent: goodbye
  - action: utter_goodbye
```

```
forms:
  flight_form:
    required_slots:
      depart_date.day_number:
        - entity: depart_date.day_number
          type: from_entity
      depart_date.month_name:
        - entity: depart_date.month_name
          type: from_entity
      fromloc.city_name:
        - entity: fromloc.city_name
          type: from_entity
      toloc.city_name:
        - entity: toloc.city_name
          type: from_entity
```

# Rules

```
- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye
```

```
- rule: Greet everytime the user greets
  steps:
  - intent: greet
  - action: utter_greet
```



Dialog-flow



# Q&A

```
- story: mealComAirline
steps:
- intent: meal
  entities:
  - airline_name: ''
  - meal: ''
- action: action_qa
```

```
- story: mealComCidades
steps:
- intent: meal
  entities:
  - fromloc.city_name: ''
  - toloc.city_name: ''
  - meal: ''
- action: action_qa
```

```
- story: capacityComAirlineCode
steps:
- intent: capacity
  entities:
  - airline_code: ''
- action: action_qa
```

```
- story: ground_serviceComFromLoc
steps:
- intent: ground_service
  entities:
  - fromloc.city_name: ''
- action: action_qa
```

```
- story: airline
steps:
- intent: airline
  entities:
  - fromloc.city_name: ''
  - toloc.city_name: ''
- action: action_qa
```

```
- story: ground_fareComCity
steps:
- intent: ground_fare
  entities:
  - city_name: ''
- action: action_qa
```

```
- story: capacityComAirline
steps:
- intent: capacity
  entities:
  - airline_name: ''
- action: action_qa
```

```
- story: ground_fareComAirportName
steps:
- intent: ground_fare
  entities:
  - fromloc.airport_name: ''
- action: action_qa
```

```
- story: aircraft
steps:
- intent: aircraft
  entities:
  - fromloc.city_name: ''
  - toloc.city_name: ''
- action: action_qa
```

```
- story: quantity
steps:
- intent: quantity
  entities:
  - fromloc.city_name: ''
  - toloc.city_name: ''
- action: action_qa
```

```
- story: airport
steps:
- intent: airport
  entities:
  - city_name: ''
- action: action_qa
```

```
- story: distance
steps:
- intent: distance
  entities:
  - fromloc.city_name: ''
  - toloc.city_name: ''
- action: action_qa
```

```
- story: ground_serviceComCity
steps:
- intent: ground_service
  entities:
  - city_name: ''
- action: action_qa
```

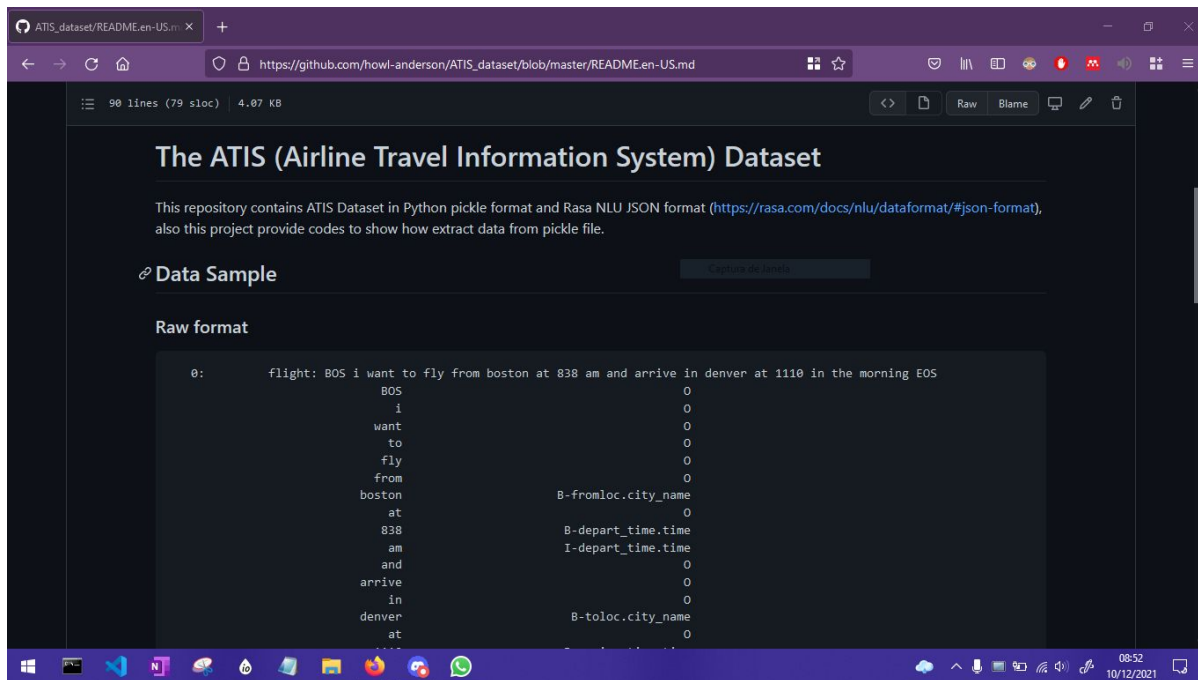
```
- story: restriction
steps:
- intent: restriction
  entities:
  - fromloc.city_name: ''
  - toloc.city_name: ''
- action: action_qa
```

Stories utilizadas no treinamento

Stories usados no teste

# Busca

# Dados usados



90 lines (79 sloc) | 4.07 KB

## The ATIS (Airline Travel Information System) Dataset

This repository contains ATIS Dataset in Python pickle format and Rasa NLU JSON format (<https://rasa.com/docs/nlu/dataformat/#json-format>), also this project provide codes to show how extract data from pickle file.

### Data Sample

Capture de Janela

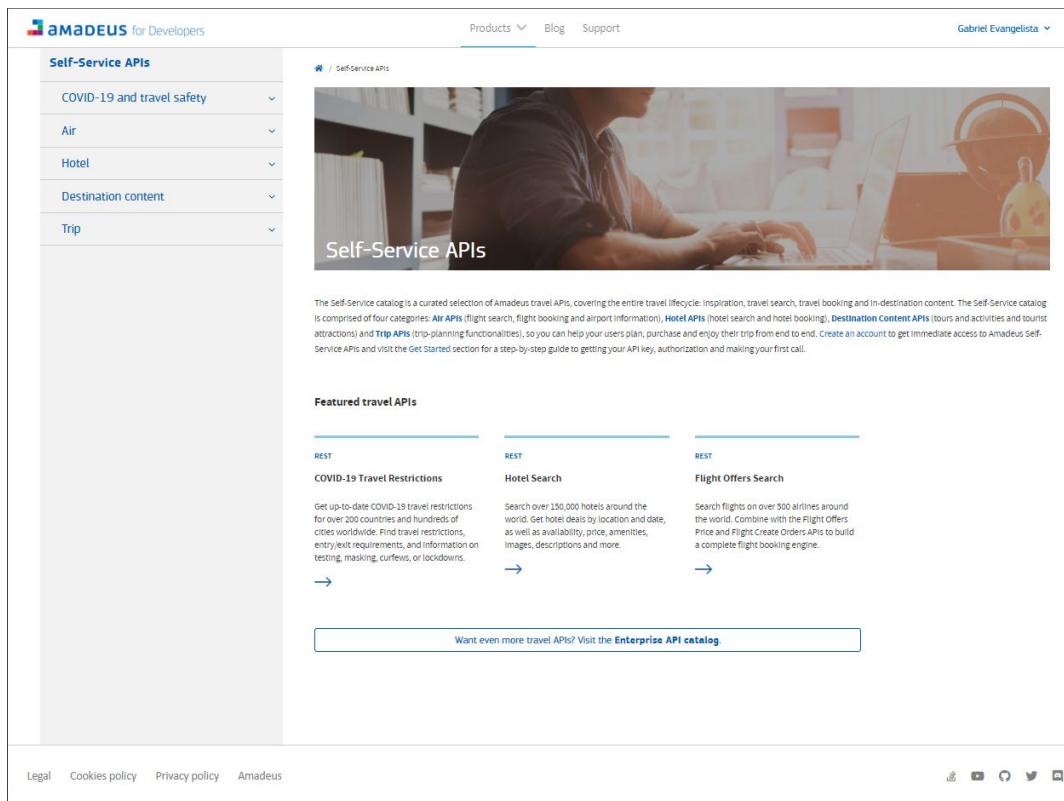
#### Raw format

```
0:      flight: BOS i want to fly from boston at 838 am and arrive in denver at 1110 in the morning EOS
      BOS                                0
      i                                  0
      want                               0
      to                                 0
      fly                                0
      from                               0
      boston                            B-fromloc.city_name
      at                                 0
      838                                B-depart_time.time
      am                                 I-depart_time.time
      and                               0
      arrive                             0
      in                                 0
      denver                             B-toloc.city_name
      at                                 0
```

[https://github.com/howl-anderson/ATIS\\_dataset](https://github.com/howl-anderson/ATIS_dataset)

# Ferramenta

- Amadeus API



The screenshot displays the Amadeus for Developers website. At the top right, there are logos for Centro de Informática UFPE and Universidade Federal de Pernambuco. The main navigation bar includes links for Products, Blog, and Support, along with a user profile for Gabriel Evangelista. A left-hand sidebar lists categories under 'Self-Service APIs': COVID-19 and travel safety, Air, Hotel, Destination content, and Trip. The main content area features a hero image of a person working on a laptop with the title 'Self-Service APIs'. Below this, a paragraph explains the catalog's scope and categories: Air APIs, Hotel APIs, Destination Content APIs, and Trip APIs. A 'Featured travel APIs' section follows, highlighting three REST APIs: COVID-19 Travel Restrictions, Hotel Search, and Flight Offers Search, each with a brief description and a link. At the bottom, a banner encourages visiting the Enterprise API catalog. The footer contains legal links and social media icons.

amadeus for Developers

Products Blog Support Gabriel Evangelista

Self-Service APIs

COVID-19 and travel safety Air Hotel Destination content Trip

Self-Service APIs

The Self-Service catalog is a curated selection of Amadeus travel APIs, covering the entire travel lifecycle: inspiration, travel search, travel booking and in-destination content. The Self-Service catalog is comprised of four categories: **Air APIs** (flight search, flight booking and airport information), **Hotel APIs** (hotel search and hotel booking), **Destination Content APIs** (tours and activities and tourist attractions) and **Trip APIs** (trip-planning functionalities), so you can help your users plan, purchase and enjoy their trip from end to end. Create an account to get immediate access to Amadeus Self-Service APIs and visit the [Get Started](#) section for a step-by-step guide to getting your API key, authorization and making your first call.

Featured travel APIs

**COVID-19 Travel Restrictions**  
REST  
Get up-to-date COVID-19 travel restrictions for over 200 countries and hundreds of cities worldwide. Find travel restrictions, entry/exit requirements, and information on testing, masking, curfews, or lockdowns.  
→

**Hotel Search**  
REST  
Search over 150,000 hotels around the world. Get hotel deals by location and date, as well as availability, price, amenities, images, descriptions and more.  
→

**Flight Offers Search**  
REST  
Search flights on over 500 airlines around the world. Combine with the Flight Offers Price and Flight Create Orders APIs to build a complete flight booking engine.  
→

Want even more travel APIs? Visit the [Enterprise API catalog](#)

Legal Cookies policy Privacy policy Amadeus

<https://developers.amadeus.com/>

cin.ufpe.br



# Integração

- Obtenção de Token
- Chamada para conversão de nome para IATA code
- Chamada final para obtenção dos voos entre as cidades e na data especificados

```
headers = {
    'Content-Type': 'application/x-www-form-urlencoded',
}

data = {
    'grant_type': 'client_credentials',
    'client_id': apiKey,
    'client_secret': apiSecret
}

response = requests.post('https://test.api.amadeus.com/v1/security/oauth2/token', headers=headers, data=data)
dicti = json.loads(response.text)
```

```
headers = {
    'Authorization': 'Bearer ' + dicti['access_token'],
}

params = (
    ('subType', 'CITY'),
    ('keyword', str(tracker.get_slot('fromloc.city_name'))),
)

response2 = requests.get('https://test.api.amadeus.com/v1/reference-data/locations', headers=headers, params=params)

fromCity = json.loads(response2.text)['data'][0]['iataCode']
```

```
params = (
    ('originLocationCode', str(fromCity)),
    ('destinationLocationCode', str(toCity)),
    ('departureDate', str(dataVoo.strftime('%Y-%m-%d'))),
    ('adults', 1),
)

response2 = requests.get('https://test.api.amadeus.com/v2/shopping/flight-offers', headers=headers, params=params)

dicti2 = json.loads(response2.text)
```