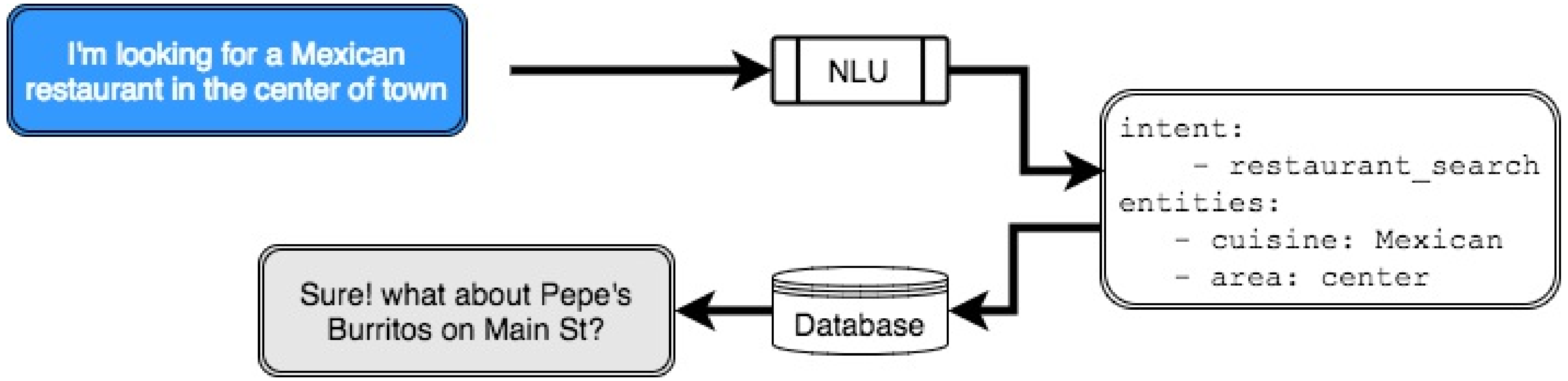
BUILDING CHATBOTS IN PYTHON

Understanding intents and entities

Alan Nichol
Co-founder and CTO, Rasa

An example


I'm looking for a Mexican restaurant in the center of town
NLU
Database
Sure! what about Pepe's Burritos on Main St?

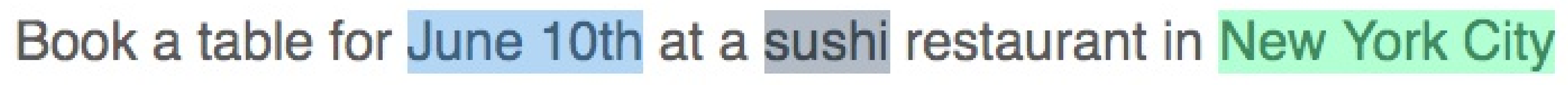
```
intent:
    - restaurant_search
entities:
   - cuisine: Mexican
   - area: center
```

Intents

A `restaurant_search` can be expressed many different ways:

- I'm hungry
- Show me good pizza spots
- I want to take my boyfriend out for sushi
    - Can also be `request_booking`

Entities


Book a table for June 10th at a sushi restaurant in New York City

- NER = Named Entity Recognition

Regular expressions to recognize intents and exercises

- Simpler than machine learning approaches
- Highly computationally efficient
- Drawback:
    - Debugging regular expressions can become difficult

Using regular expressions

- '|' is equivalent to OR
- \b matches the beginning or end of a word

```
In [1]: re.search(r"(hello|hey|hi)", "hey there!") is not None
Out[1]: True

In [2]: re.search(r"(hello|hey|hi)", "which one?") is not None
Out[2]: True

In [3]: re.search(r"\b(hello|hey|hi)\b", "hey there!") is not None
Out[3]: True

In [4]: re.search(r"\b(hello|hey|hi)\b", "which one?") is not None
Out[4]: False
```

Using regex for entity recognition

```
In [1]: pattern = re.compile('[A-Z]{1}[a-z]*')

In [2]: message = """
Mary is a friend of mine,
she studied at Oxford and
now works at Google"""

In [3]: pattern.findall(message)
Out[3]: ['Mary', 'Oxford', 'Google']
```

BUILDING CHATBOTS IN PYTHON

Let's practice!

BUILDING CHATBOTS IN PYTHON

Word vectors

Alan Nichol
Co-founder and CTO, Rasa

Machine learning

- Programs which can get better at a task by being exposed to more data
- Identifying which intent a user message belongs to

Vector representations

"can you help me please?"

Units	examples	vectors
characters	"c", "a", "n", ...	v_c, v_a, v_n, ...
words	"can", "you", ...	v_{can}, v_{you}, ...
sentences	"can you help..."	v_{can you help ...}

Word vectors

Context	Candidates
let's meet at the ___ tomorrow	office, gym, park, beach, party
I love going to the ___ to play with the dogs	beach, park

- Word vectors try to represent *meaning* of words
- Words which appear in similar context have similar vectors

Word vectors are computationally intensive

- Training word vectors requires a lot of data
- High quality word vectors are available for anyone to use
- GloVe algorithm
  - Cousin of word2vec
- spaCy

Word vectors in spaCy

```
In [1]: import spacy

In [2]: nlp = spacy.load('en')

In [3]: nlp.vocab.vectors_length
Out[3]: 300

In [4]: doc = nlp('hello can you help me?')

In [5]: for token in doc:
   ...:     print("{} : {}".format(token, token.vector[:3]))
hello : [ 0.25233001  0.10176    -0.67484999]
can : [-0.23857     0.35457    -0.30219001]
you : [-0.11076     0.30785999 -0.51980001]
help : [-0.29370001  0.32253    -0.44779   ]
me : [-0.15396     0.31894001 -0.54887998]
? : [-0.086864    0.19160999  0.10915   ]
```

Similarity

- Direction of vectors matters
- "Distance" between words = angle between the vectors
- Cosine similarity
  - 1: If vectors point in the same direction
  - 0: If they are perpendicular
  - -1: If they point in opposite directions

.similarity()

- "can" and "cat" are spelled similarly but have low similarity
- but "cat" and "dog" have high similarity

```
In [1]: import spacy

In [2]: nlp = spacy.load('en')

In [3]: doc = nlp("cat")

In [4]: doc.similarity(nlp("can"))
Out[4]: 0.30165292161215396

In [5]: doc.similarity(nlp("dog"))
Out[5]: 0.8016855173294953
```

BUILDING CHATBOTS IN PYTHON

Let's practice!

BUILDING CHATBOTS IN PYTHON

Intents and classification

Alan Nichol
Co-founder and CTO, Rasa

Supervised learning

- A classifier predicts the intent label given a sentence
- 'Fit' classifier by tuning it on *training data*
- Evaluate performance on *test data*
- Accuracy: the fraction of labels we predict correctly

ATIS dataset

- Thousands of sentences with labeled intents and entities
- Collected from a real flight booking service
- Intents like
  - atis_flight
  - atis_airfare

ATIS dataset II

```
In [1]: sentences_train[:2]
Out[1]: [
  "i want to fly from boston at 838 am
  and arrive in denver at 1110 in the morning",
  "what flights are available from pittsburgh
  to baltimore on thursday morning"
]

In [2]: labels_train[:2]
Out[2]: [
  "atis_flight",
  "atis_flight"
]

In [3]: import numpy as np

In [4]: X_train_shape = (len(sentences_train),nlp.vocab.vectors_length)

In [5]: X_train = np.zeros(X_train_shape)

In [6]: for sentence in sentences_train:
   ...:     X_train[i,:] = nlp(sentence).vector
```

Nearest neighbor classification

- Need training data
  - Sentences which we've already labeled with their intents
- Simplest solution:
  - Look for the labeled example that's most similar
  - Use its intent as a best guess
- Nearest neighbor classification

Nearest neighbor classification in scikit-learn

```
In [1]: from sklearn.metrics.pairwise import cosine_similarity

In [2]: test_message = """
i would like to find a flight from charlotte
to las vegas that makes a stop in st. louis"""

In [3]: test_x = nlp(test_message).vector

In [4]: scores = [
   ...:     cosine_similarity(X[i,:], test_x)
   ...:     for i in range(len(sentences_train)
   ...: ]

In [5]: labels_train[np.argmax(scores)]
Out[5]: 'atis_flight'
```

Support vector machines

- Nearest neighbours is very simple - we can do better
- SVM / SVC: support vector machine / classifier

```
In [1]: from sklearn.svm import SVC

In [2]: clf = SVC()

In [3]: clf.fit(X_train, y_train)

In [4]: y_pred = clf.predict(X_test)
```

BUILDING CHATBOTS IN PYTHON

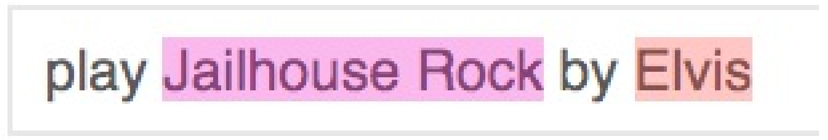
Let's practice!

BUILDING CHATBOTS IN PYTHON

Entity extraction

Alan Nichol
Co-founder and CTO, Rasa

Beyond keywords: Context

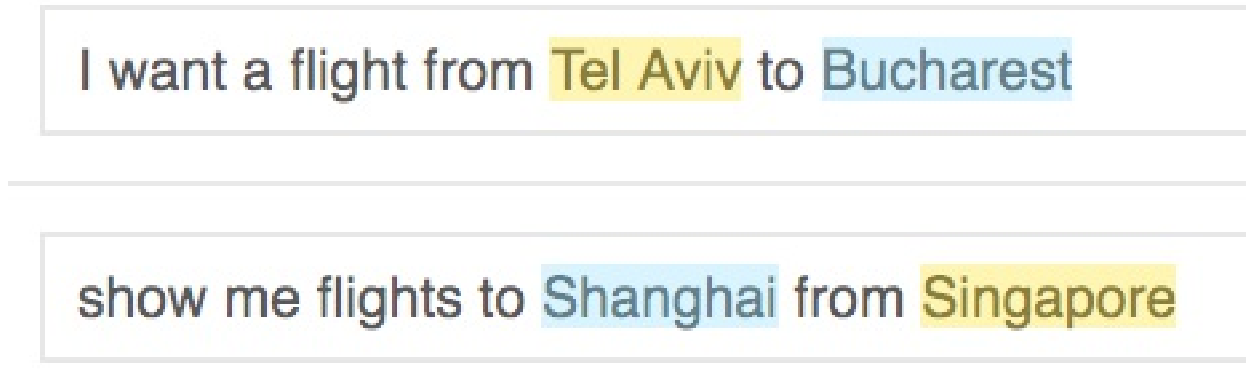

play Jailhouse Rock by Elvis

- Keywords don't work for entities you haven't seen before
- Use contextual clues:
  - Spelling
  - Capitalization
  - Words occurring before & after
- Pattern recognition

Pre-built Named Entity Recognition

```
In [1]: import spacy

In [2]: nlp = spacy.load('en')

In [3]: doc = nlp("my friend Mary has worked at Google since 2009")

In [4]: for ent in doc.ents:
   ...:     print(ent.text, ent.label_)
   ...:
Mary PERSON
Google ORG
2009 DATE
```

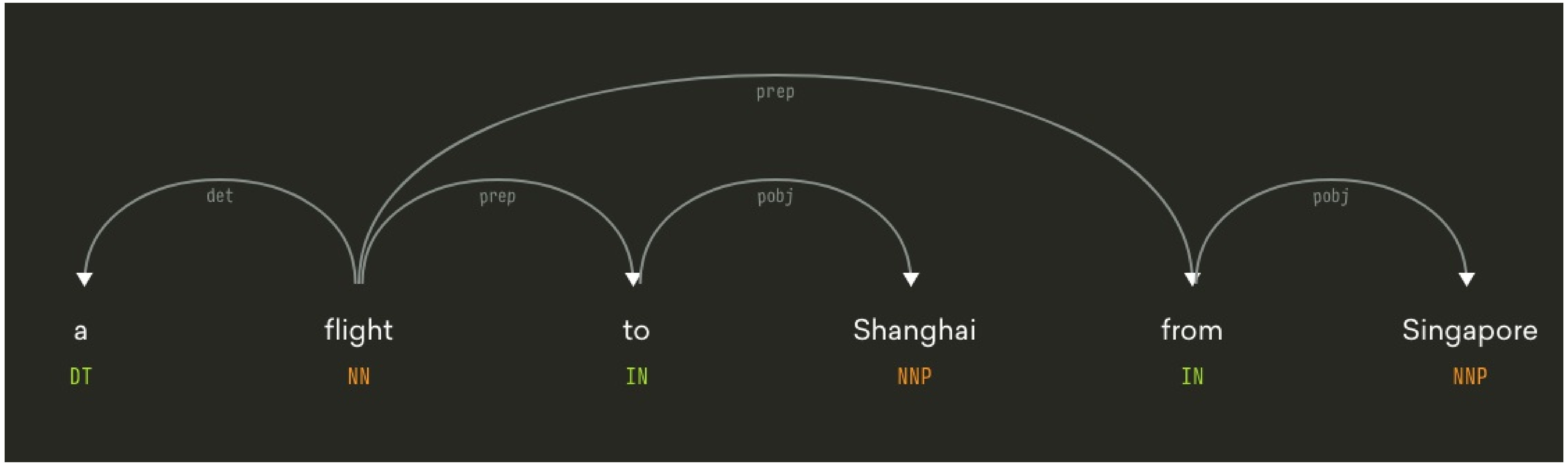
Roles


I want a flight from Tel Aviv to Bucharest

show me flights to Shanghai from Singapore

```
In [1]: pattern_1 = re.compile('.* from (.*) to (.*)')

In [2]: pattern_2 = re.compile('.* to (.*) from (.*)')
```

Dependency parsing



```
In [1]: doc = nlp('a flight to Shanghai from Singapore')

In [2]: shanghai, singapore = doc[3], doc[5]

In [3]: list(shanghai.ancestors)
Out[3]: [from, flight]

In [4]: list(singapore.ancestors)
Out[4]: [to, flight]
```

Shopping example

```
In [1]: doc = nlp("let's see that jacket in red and some blue jeans")

In [2]: items = [doc[4], doc[10]]  # [jacket, jeans]

In [3]: colors = [doc[6], doc[9]]  # [red, blue]

In [4]: for color in colors:
    ...:     for tok in color.ancestors:
    ...:         if tok in items:
    ...:             print("color {} belongs to item {}".format(color, tok))
    ...:             break
color red belongs to item jacket
color blue belongs to item jeans
```

BUILDING CHATBOTS IN PYTHON

Let's practice!

BUILDING CHATBOTS IN PYTHON

Robust NLU with Rasa

Alan Nichol
Co-founder and CTO, Rasa

Rasa NLU

- Library for intent recognition & entity extraction
- Based on spaCy, scikit-learn, & other libraries
- Built in support for chatbot specific tasks

Rasa data format

```
In [1]: from rasa_nlu.converters import load_data

In [2]: training_data = load_data("./training_data.json")

In [3]: import json

In [4]: print(json.dumps(data.training_examples[22], indent=2))
Out[4]: {
  "text": "i'm looking for a place in the north of town",
  "intent": "restaurant_search",
  "entities": [
    {
      "start": 31,
      "end": 36,
      "value": "north",
      "entity": "location"
    }
  ]
}
```

Interpreters

```
In [1]: message = "I want to book a flight to London"

In [2]: interpreter.parse(message))
Out[2]: {
  "intent": {
    "name": "flight_search",
    "confidence": 0.9
  },
  "entities": [
    {
      "entity": "location",
      "value": "London",
      "start": 27,
      "end": 33
    }
  ]
}
```

Rasa usage

```
# Creating a model
In [1]: from rasa_nlu.config import RasaNLUConfig

In [2]: from rasa_nlu.model import Trainer

In [3]: config = RasaNLUConfig(cmdline_args={"pipeline": "spacy_sklearn"})

In [4]: trainer = Trainer(config)

In [5]: interpreter = trainer.train(training_data)
```

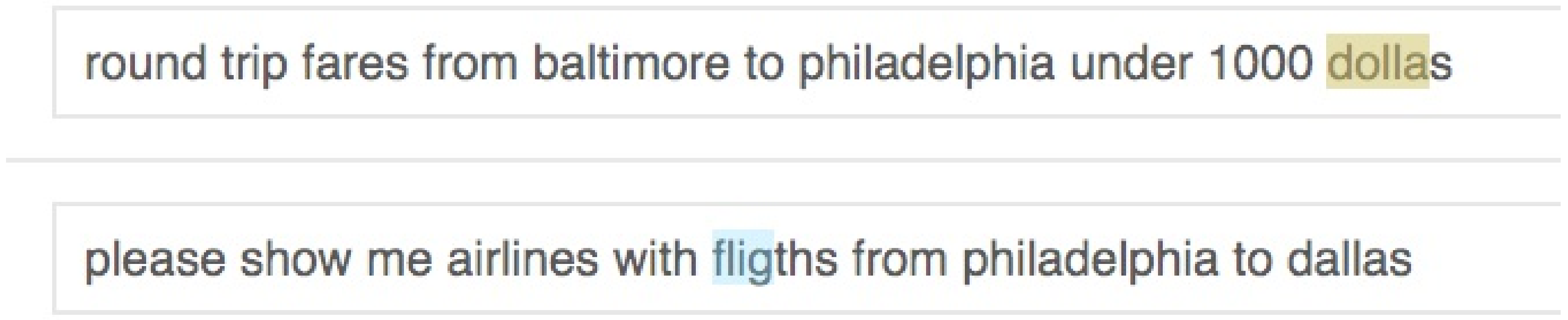
Rasa pipelines

```
In [1]: spacy_sklearn_pipeline = [
  "nlp_spacy",
  "ner_crf",
  "ner_synonyms",
  "intent_featurizer_spacy",
  "intent_classifier_sklearn"
]

# These two statements are identical:
In [2]: RasaNLUConfig(
            cmdline_args={"pipeline": spacy_sklearn_pipeline}
        )
Out[2]: <rasa_nlu.config.RasaNLUConfig at 0x10f60aa90>

In [3]: RasaNLUConfig(
            cmdline_args={"pipeline": "spacy_sklearn"}
        )
Out[3]: <rasa_nlu.config.RasaNLUConfig at 0x10f60aa20>
```

Conditional random fields

- Machine Learning model, popular for named entity recognition
    - can perform well even with small training data

Handling typos


round trip fares from baltimore to philadelphia under 1000 dollas

please show me airlines with fligths from philadelphia to dallas

```
In [1]: pipeline = [
   ...:     "nlp_spacy",
   ...:     "intent_featurizer_spacy",
   ...:     "intent_featurizer_ngrams",
   ...:     "intent_classifier_sklearn"
   ...: ]
```

BUILDING CHATBOTS IN PYTHON

Let's practice!