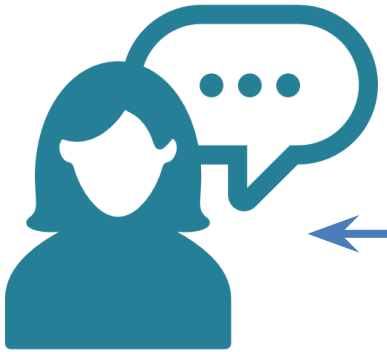
DeepPavlov:
Goal-oriented dialog system

Mariia Vikhreva
Laboratory of Neural Systems and Deep Learning

Bot architecture



Is it windy on weekends?

textual data

Natural Language Understanding - NLU
- Domain detection
- Intent classification
- Entity recognition

domain = weather_service
intent = request_weather
entities = { subtype = 'windy',
date = 'weekends ' }

semantic frame

Dialogue Manager - DM
- Dialog state tracking
- Action policy managing

action = request_location

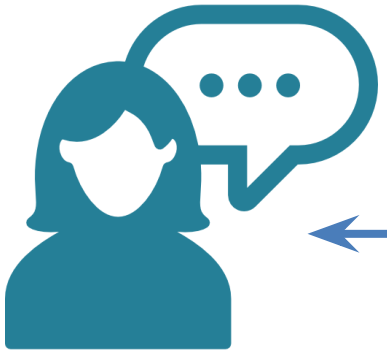
bot action

Natural Language Generation - NLG
- Generative models
- Templates

Where are you?

textual data

Bot architecture



Is it windy on weekends?

textual data

Named Entity Recognizer (NER)

Natural Language Understanding - NLU

- Domain detection
- Intent classification
- Entity recognition

domain = weather_service
intent = request_weather
entities = { subtype = 'windy',
date = 'weekends ' }

semantic frame

Where are you?

textual data

Natural Language Generation - NLG

- Generative models
- Templates

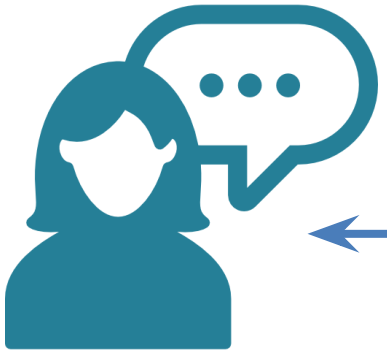
action = request_location

bot action

Dialogue Manager - DM

- Dialog state tracking
- Action policy managing

Bot architecture



Is it windy on weekends?

textual data

Named Entity Recognizer (NER)

Natural Language Understanding - NLU

- Domain detection
- Intent classification
- Entity recognition

domain = weather_service
intent = request_weather
entities = { subtype = 'windy',
date = 'weekends ' }

semantic frame

Recurrent Neural Network (RNN)

action = request_location

bot action

Dialogue Manager - DM

- Dialog state tracking
- Action policy managing

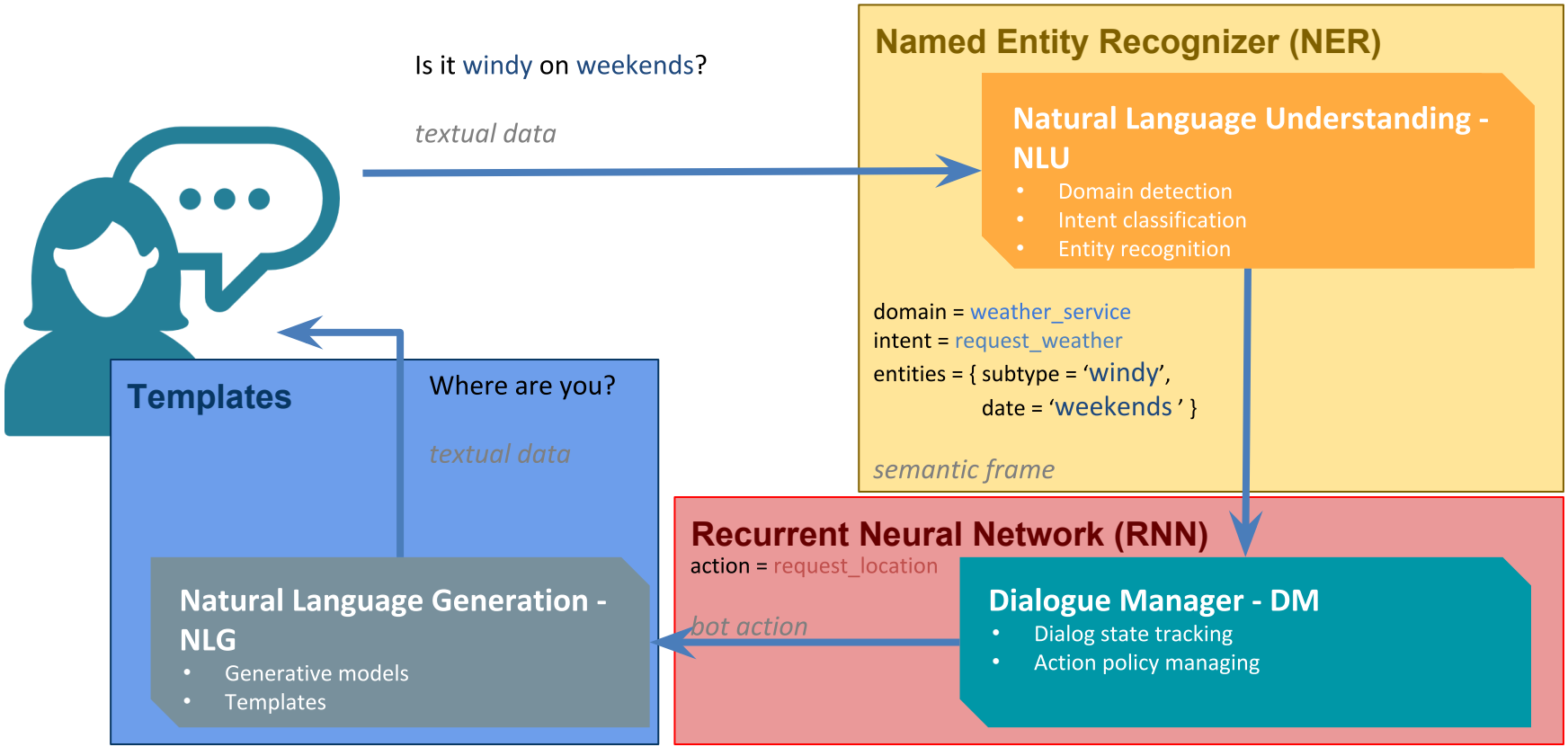
Natural Language Generation - NLG

- Generative models
- Templates

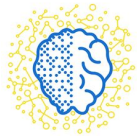
Where are you?

textual data

Bot architecture



Dataset Reader

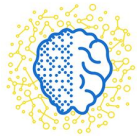


- Dataset Reader is responsible for downloading dataset

```
from deeppavlov.dataset_readers.dstc2_reader import DSTC2Version2DatasetReader

data = DSTC2Version2DatasetReader().read(data_path="tmp/my_download_of_dstc2")
```

- DSTC2 (Dialogue State Tracking Challenge 2) data is now
  - downloaded from web
  - saved to `./tmp/my_download_of_dstc2`

Dataset Iterator

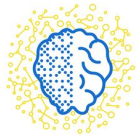


- Dataset Iterator is responsible for generating batches

```
from deeppavlov.dataset_iterators.dialog_iterator import DialogDatasetIterator

batches_generator = DialogDatasetIterator(data, seed=1443, shuffle=True)\
                                          .gen_batches(batch_size=4)
```

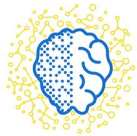
- `batches_generator` is iterator over data batches

What's in a batch?



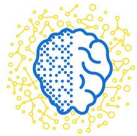
Speaker 1 (human)		Speaker 2 (future bot)	
'text'	'I want cheap restaurant in the north of town.'	'text'	'Dodo Pizza is a nice restaurant, their phone number is 8(800)333-00-60.'
'slots'	{'pricerange': 'cheap', 'location': 'north'}		
'intents' *(optional)*	[inform_pricerange, inform_location, request_restaurant]	'act'	[inform_restaurant, inform_phone]
		'db_result' *(optional)*	{'name': 'Dodo Pizza', 'pricerange': 'cheap', 'location': 'north', 'cuisine': 'italian', 'phone': '8(800)333-00-60'}

Extra data: Templates



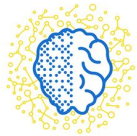
Response templates for Speaker 2	
mapping action → text	welcome_msg → Hello, welcome to the Cambridge restaurant system. You can ask for restaurants by area, price range or food type. How may I help you? inform_place → The #name restaurant is on #address.

HowTo: Configs



- **`dataset_reader`** — configuration of dataset reader component
  - data download and saving to disk
- **`dataset_iterator`** — configuration of dataset iterator component
  - generator of batches
- **`metadata`** — extra info
  - urls for extra data download
  - telegram configuration
- **`train`** — training process configuration
  - size of batches
  - number of training epochs
- **`chainer`** specifies data flow
  - which components are run and in what order

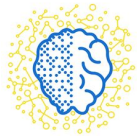
Vocab config



Let’s construct a vocabulary that:

1. Takes utterances of speaker 1
2. Splits them into tokens
3. Builds a dictionary of all tokens
4. Outputs index for an input token

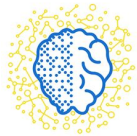
Vocab config



- downloads DSTC2 data files
- saves to `DEEPPAVLOV_ROOT/../download/dstc2_v2`

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'}}
```

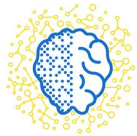
Vocab config



- inputs data from ‘data_reader’
- generates batches (“groups” of data samples)

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'}}
```

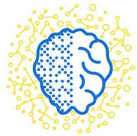
Vocab config



- specifies urls for required data download

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]}}
```

Vocab config



- parameters during training phase
- empty, because vocab *doesn't need training*

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {}
}
```

Vocab config

- specifies data flow

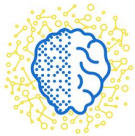
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab config

- name of input variables

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab config



- name of input variables *available during training*

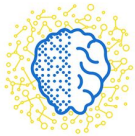
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab config

- predicted variables

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab config



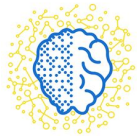
- consequently run components
- consists of one `default_vocab` component

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Components

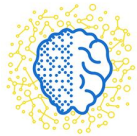
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Components: *required parameters*



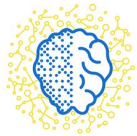
- **`name`** — registered name of the component
  - it is a link to python component implementation
- **`save_path`** — path to save the component
  - sometimes is optional, for example, for tokenizers
- **`load_path`** — path to load the component
  - sometimes is optional, for example, for tokenizers

Components: *optional parameters*



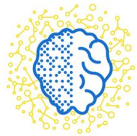
- **`id`** — reference name for a component
- **`ref`** — "id" of a component that was previously initialized
  - ➢ can be used instead of **`name`** parameter
- **`fit_on`** — a list of data fields to fit on
  - ➢ calls __fit__ method of the component
- **`in`** — input variables during inference
- **`out`** — output variables during inference

Components: *other parameters*



Components might have their own unique parameters.

Vocab component config



- name of the component

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab component config

- path to load component from

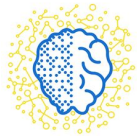
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab component config

- path to save component to

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

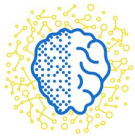
Vocab component config



- fit (build) on whole dataset once
- take only ‘utterance’ variable from data

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

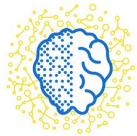
Vocab component config



- input variable during inference
- `default_vocab` inputs tokens

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab component config



- output variable during inference
- `default_vocab` outputs indices for input tokens

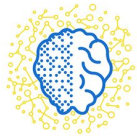
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab component config

- ‘default_vocab’ specific parameter
- build vocabulary of tokens (character level is also available)

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

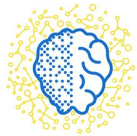
Vocab component config



- ‘default_vocab’ specific parameter
- use the ‘split_tokenizer’ component to get tokens from text

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [{'subdir': 'dstc2_v2',
                            'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'}]},
 'train': {},
 'chainer': {'in': ['utterance'],
             'in_y': [],
             'out': ['utterance_token_indices'],
             'pipe': [{'name': 'default_vocab',
                       'load_path': 'vocabs/token.dict',
                       'save_path': 'vocabs/token.dict',
                       'fit_on': ['utterance'],
                       'in': ['utterance'],
                       'out': ['utterance_token_indices'],
                       'level': 'token',
                       'tokenizer': {'name': 'split_tokenizer'}}]}}
```

Vocab build



- Saving config to disk

```
json.dump(vocab_config, open('gobot/vocab_config.json', 'wt'))
```

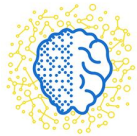
- Downloading data required for building

```
from deeppavlov.download import deep_download

deep_download(['--config', 'gobot/vocab_config.json'])
```

- Building

```
from deeppavlov.core.commands.train import train_evaluate_model_from_config

train_evaluate_model_from_config('gobot/vocab_config.json')
```

Vocab use



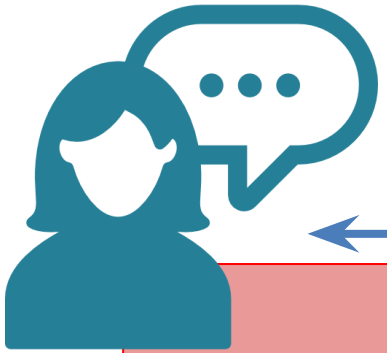
- Initializing vocab

```
from deeppavlov.core.commands.infer import build_model_from_config

vocab = build_model_from_config('gobot/vocab_config.json')
```

- Calling

```
vocab(['hi'])
> 141
```

Bot architecture



Is it windy on weekends?

textual data

Named Entity Recognizer (NER)

Natural Language Understanding - NLU

- Domain detection
- Intent classification
- Entity recognition

domain = weather_service
intent = request_weather
entities = { subtype = 'windy',
date = 'weekends ' }

semantic frame

Where are you?

textual data

THE TUTORIAL

Natural Language Generation - NLG

- Generative models
- Templates

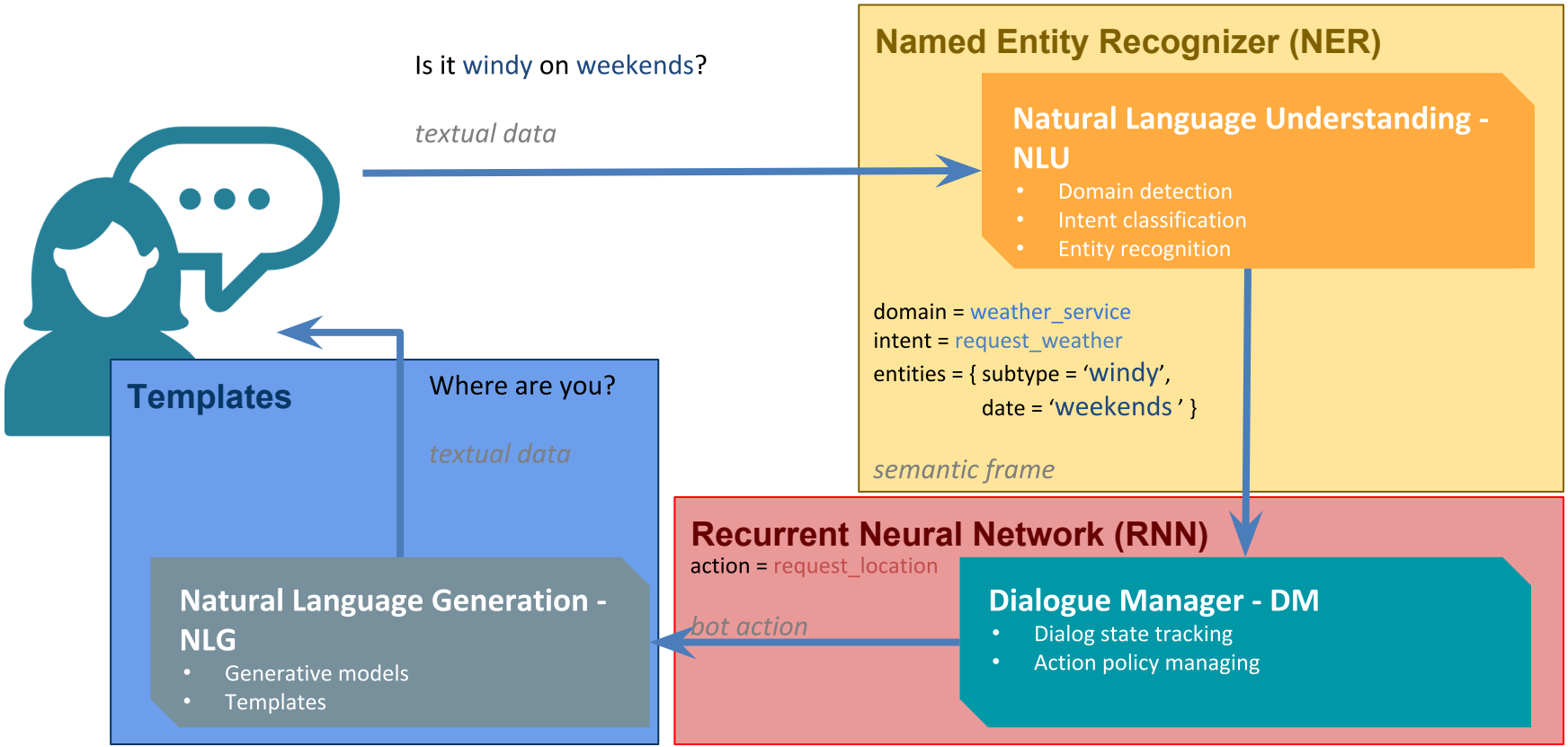
action = request_location

bot action

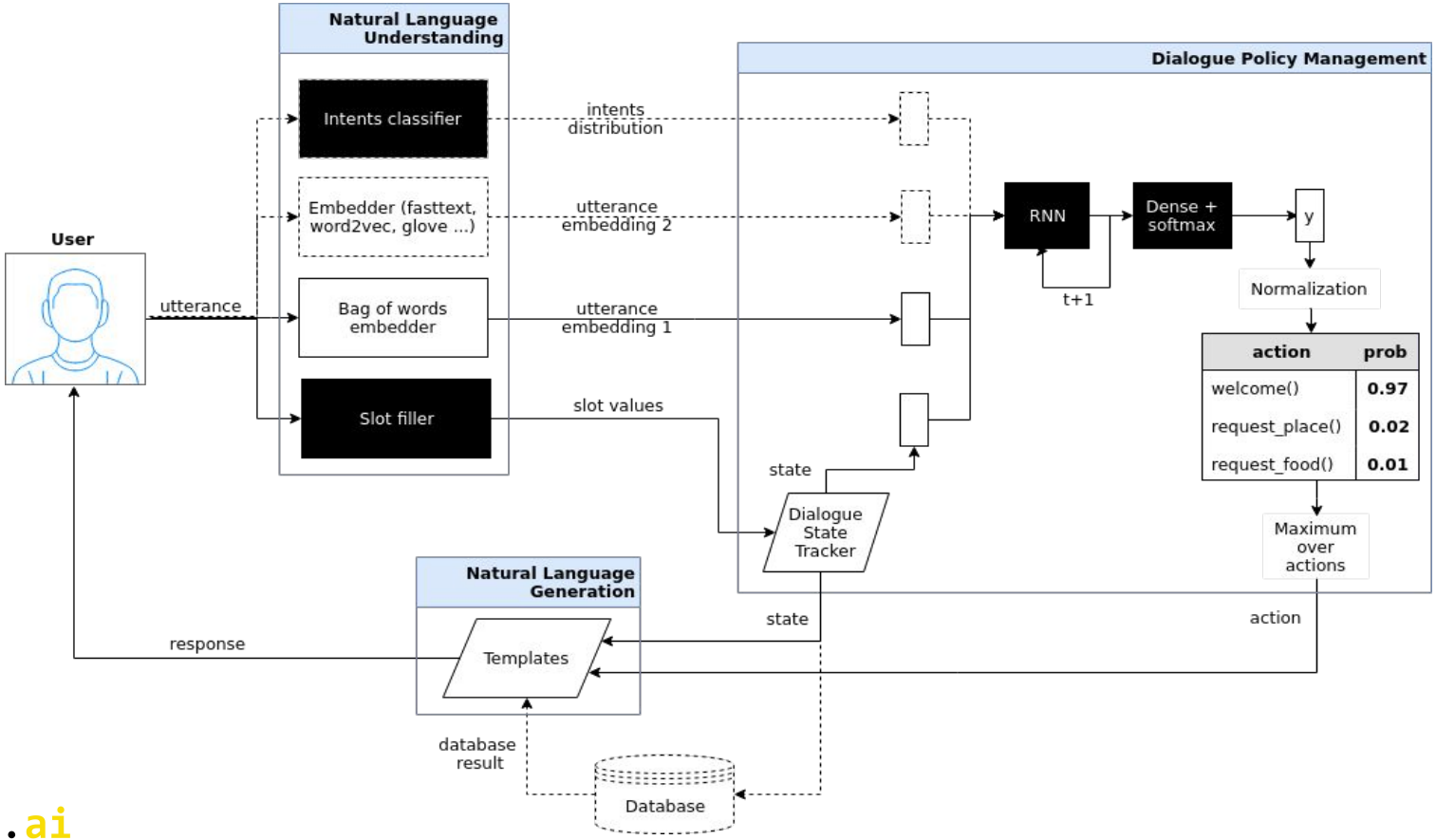
Dialogue Manager - DM

- Dialog state tracking
- Action policy managing

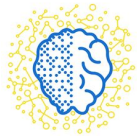
Bot architecture



Bot architecture



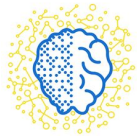
Bot config



- same ‘dataset_reader’

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'}}
```

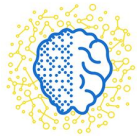
Bot config



- same ‘dataset_iterator’

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'}}
```

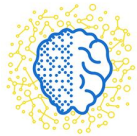
Bot config



- ‘metadata.download’ contains the same ‘dstc2_v2’ url
- and download url for fasttext embeddings

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]}}
```

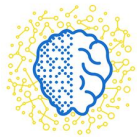
Bot config



- parameters for training phase

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

Bot config



- number of samples in a batch

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

Bot config

- number of epochs (epoch — single run on the whole dataset)

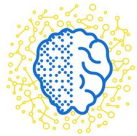
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

Bot config

- logging parameters (do logging every epoch)

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

Bot config



- metrics used for evaluation
- ‘per_item_dialog_accuracy’ is accuracy of response tokens

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

Bot config

- validation parameters (calculate metrics on ‘valid’ dataset every epoch)

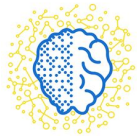
```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

Bot config

- we are able to endure 20 epochs without metric improvement on ‘valid’ data
- after 20 epochs training is finished

```
{'dataset_reader': {'name': 'dstc2_v2_reader', 'data_path': 'dstc2_v2'},
 'dataset_iterator': {'name': 'dialog_iterator'},
 'metadata': {'download': [
    {'subdir': 'dstc2_v2',
     'url':'http://lnsigo.mipt.ru/export/datasets/dstc2_v2.tar.gz'},
    {'subdir': 'embeddings',
     'url':'http://lnsigo.mipt.ru/export/deeppavlov_data/embeddings/dstc2_fastText_model.bin'}
 ]},
 'train': {'batch_size': 4,
           'epochs': 2,
           'log_every_n_batches': -1,
           'log_every_n_epochs': 1,
           'metrics': ['per_item_dialog_accuracy'],
           'val_every_n_epochs': 1,
           'validation_patience': 20}}
```

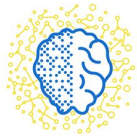
Bot config



- chainer takes as input ‘x’ — dict with user utterance info, ‘y’ — dict with response info,
- and outputs ‘y_predicted’ — predicted textual response

```
'chainer': {'in': ['x'],
            'in_y': ['y'],
            'out': ['y_predicted']}}
```

Bot config



- vocabulary component

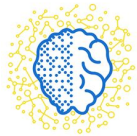
```
'chainer': {'in': ['x'],
            'in_y': ['y'],
            'out': ['y_predicted'],
            'pipe': [{'name': 'default_vocab',
                      'id': 'token_vocab',
                      'load_path': 'vocabs/token.dict',
                      'save_path': 'vocabs/token.dict',
                      'fit_on': ['x'],
                      'level': 'token',
                      'tokenizer': {'name': 'split_tokenizer'}},
                     {'name': 'sqlite_database',
                      'id': 'restaurant_database',
                      'save_path': 'dstc2_v2/resto.sqlite',
                      'primary_keys': ['name'],
                      'table_name': 'mytable'},
```

Bot config

- database component

```
'chainer': {'in': ['x'],
            'in_y': ['y'],
            'out': ['y_predicted'],
            'pipe': [{'name': 'default_vocab',
                      'id': 'token_vocab',
                      'load_path': 'vocabs/token.dict',
                      'save_path': 'vocabs/token.dict',
                      'fit_on': ['x'],
                      'level': 'token',
                      'tokenizer': {'name': 'split_tokenizer'}},
                     {'name': 'sqlite_database',
                      'id': 'restaurant_database',
                      'save_path': 'dstc2_v2/resto.sqlite',
                      'primary_keys': ['name'],
                      'table_name': 'mytable'},
```

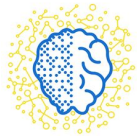
Bot config



- embedder component

```
{'dim': 100,
 'id': 'my_embedder',
 'load_path': 'embeddings/dstc2_fastText_model.bin',
 'name': 'fasttext',
 'save_path': 'embeddings/dstc2_fastText_model.bin'},
```

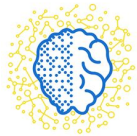
Bot config



- bot component

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

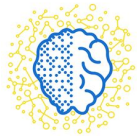
Bot config



- registered name of bot is ‘go_bot’

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

Bot config



- during inference bot takes ‘x’ (human utterance info) and predicts ‘y_predicted’ (textual response)
- during training it also takes true responses ‘y’

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

Bot config

- bot uses ‘tokenizer’ component to get tokens from human utterance

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

Bot config

- uses vocabulary with utterance tokens (‘word_vocab’) and bag-of-words embedder (‘bow_embedder’) to generate one-hot encoder of input utterance

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

Bot config

- uses dense embedder ‘embedder’ (for example, fasttext, word2vec) to embed utterance in another way

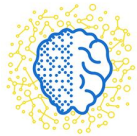
```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

Bot config

- gets slots recognized in human utterance by ‘slot_filler’ and
- updates the dialog state using state tracker ‘tracker’

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

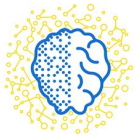
Bot config



- builds neural network with parameters ‘network_paramers’
- network expects concatenation of all embeddings as input and outputs action label (classification task)

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

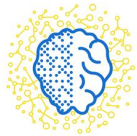
Bot config



- if action label is equal to ‘api_call_action’, then
- instead of responding bot makes an api request to database of restaurants ‘database’
- ‘database’ returns one restaurant corresponding to current dialog state

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
'api_call_action': 'api_call',
'database': '#restaurant_database',
'template_path': 'dstc2_v2/dstc2-templates.txt',
'template_type': 'DualTemplate'}]}
```

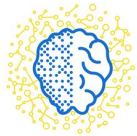
Bot config



- if action label wasn't equal to `'api_call_action'`, then
- action is mapped to a textual response using templates.
- templates are loaded from `'template_path'`

```
{'name': 'go_bot',
 'in': ['x'],
 'in_y': ['y'],
 'out': ['y_predicted'],
 'tokenizer': {'lowercase': False,
               'name': 'stream_spacy_tokenizer'},
 'word_vocab': '#token_vocab',
 'bow_embedder': {'name': 'bow'},
 'embedder': '#my_embedder',
 'slot_filler': {'config_path': 'configs/ner/slotfill_dstc2.json'},
 'tracker': {'name': 'featurized_tracker',
             'slot_names':['pricerange','this','area','food','name']},
 'network_parameters': {'load_path': 'gobot_dstc2_best/model',
                        'save_path': 'gobot_dstc2_best/model',
                        'hidden_size': 128,
                        'learning_rate': 0.002},
 'api_call_action': 'api_call',
 'database': '#restaurant_database',
 'template_path': 'dstc2_v2/dstc2-templates.txt',
 'template_type': 'DualTemplate'}]}
```

Bot config



There are even more parameters in ‘go_bot’ component, see source code for details.

Dialogue example