Conversational Agent

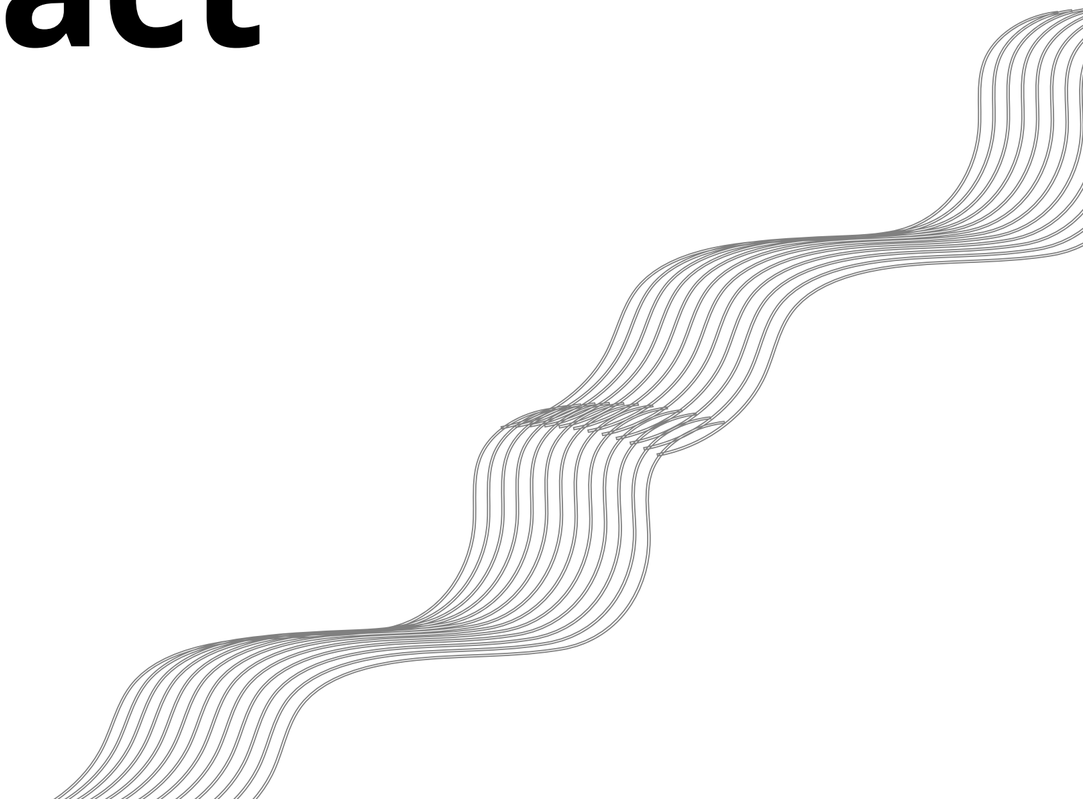
for Hotel and Restaurant Services

Ana Caicoya, Danny D. Leybzon and
Samuel García

Outline

- User dialogue act prediction with sentence embeddings
- Semantic frame slot filling with Named Entity Recognition
- Agent retrieval prediction with multioutput GB
- Agent dialogue act prediction with XGBoost
- Agent request prediction with multioutput GB
- Dialogue manager

User dialogue act prediction



Problem statement

Predict user dialogue act

Sub-problems:

- Predict “domain” (i.e. “hotel” or “restaurant”)
- Predict “act” (e.g. “Hotel-Inform”, “Restaurant-Request”, or “general-bye”)

Both are **multi-output binary classification** problems with the primary predictor: **user utterance**

Domain prediction

Initial plan: treat this as a multi-output binary classification problem with two target variables

	RESTAURANT: True	Restaurant: False
HOTEL: True	1, 1	1, 0
HOTEL: False	0, 1	0, 0

Domain prediction

Empirical analysis reveals: double-labeled acts (i.e. 1, 1) make up **less than 3%** of the test set population and **100%** of our sample are **mislabeled**

Examples:

'I would like one in the moderate price range and with free parking.',
'Where is a 4 star hotel located in North Cambridge?',
"I actually don't need reservations I just need the phone number, price range.",
'Okay. now could you help me find a restaurant in the expensive price range that is in the same area as the hotel?',
'Then find me one in the expensive price range.',
'find me a nice one and book for 5 people and 3 nights from thursday',

Domain prediction

Convert domain prediction into a **multi-class classification** problem instead of a **multi-output classification** problem by dropping examples of “both”

Build a baseline model using 'sentence-transformers/all-mpnet-base-v2' sentence embedding and xgb.XGBClassifier(objective='multi:softmax', num_class=3) Gradient Boosted Tree classifier

	precision	recall	f1-score	support
hotel	0.86	0.80	0.83	1553
none	0.83	0.88	0.85	2637
restaurant	0.80	0.77	0.79	1598
accuracy			0.83	5788
macro avg	0.83	0.82	0.82	5788
weighted avg	0.83	0.83	0.83	5788

Act prediction

Initial plan: create **two** multi-output models (based on domain prediction) with **five** target variables each

Restaurant Model: 'Restaurant-Inform', 'Restaurant-Request', 'general-thank', 'general-bye', 'general-greet'

Hotel Model: 'Hotel-Inform', 'Hotel-Request', 'general-thank', 'general-bye', 'general-greet'

Act prediction

Empirical analysis reveals: separate models, passing the domain as a variable, explicitly creating an “other” domain act, or separating “other” domain acts by service (e.g. “taxi”, “train”) has little to no impact on model performance

Hypothesized explanation: the sentence embedding intrinsically encodes this information and model is flexible enough that it’s able to learn these “implicit relationships” without being given this information explicitly

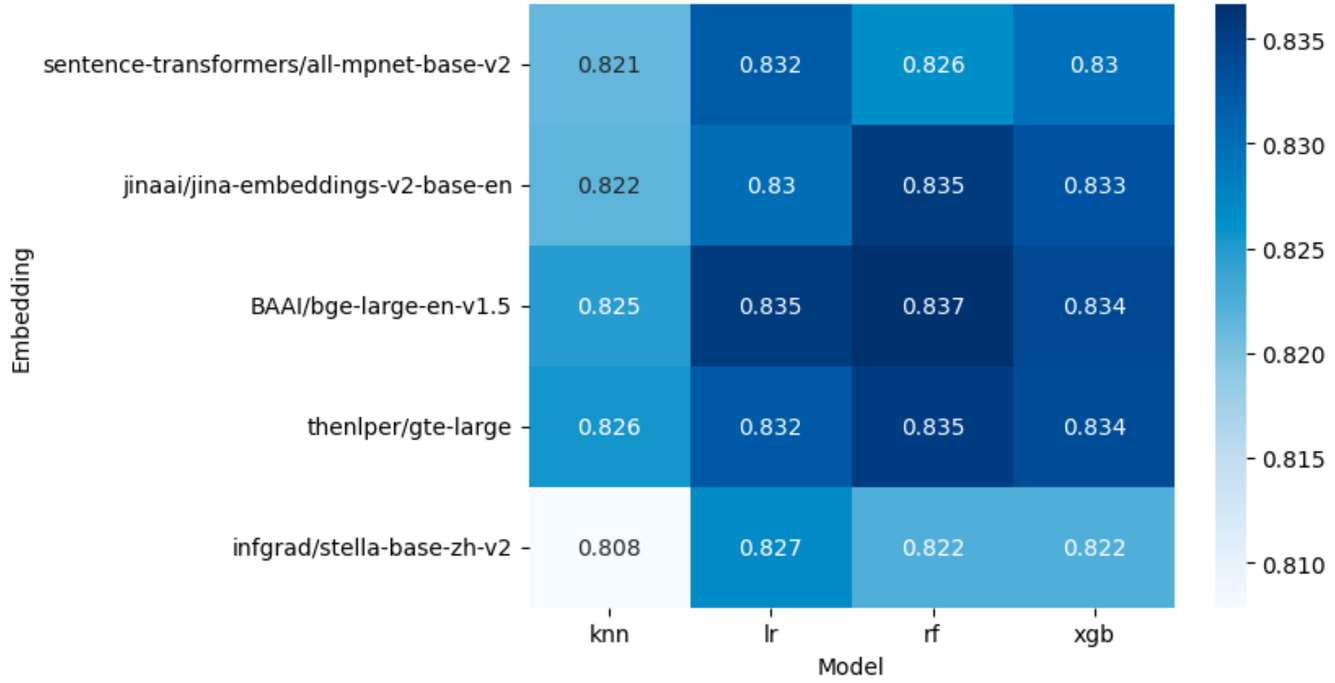
Act prediction

Build a baseline model using 'sentence-transformers/all-mpnet-base-v2' sentence embedding and MultiOutputClassifier(LogisticRegression()) multi-output logistic regression classifier from scikit-learn

	precision	recall	f1-score	support
Restaurant-Inform	0.89	0.84	0.87	1322
Hotel-Request	0.85	0.16	0.27	292
general-greet	0.00	0.00	0.00	6
Restaurant-Request	0.55	0.33	0.41	286
general-thank	0.94	0.92	0.93	693
Hotel-Inform	0.91	0.84	0.88	1328
general-bye	0.93	0.91	0.92	225
micro avg	0.89	0.77	0.83	4152
macro avg	0.73	0.57	0.61	4152
weighted avg	0.88	0.77	0.81	4152
samples avg	0.54	0.54	0.54	4152

Act prediction

Perform cross-validated grid search to find best performing embedding-model pair for act prediction. Find that ‘BAAI/bge-large-en-v1.5’ and random forest perform the best, though all models have an average F1 of above .8



BAAI/bge-large-en-v1.5

Model information:

- Takes a sentence as input; returns a length 1024 vector of floats that can be used to compare document similarity
- Pre-trained using Masked Auto-Encoder (RetroMAE)
- Fine-tuned for retrieval using contrastive learning

Meta information:

- Published by the Beijing Academy of Artificial Intelligence as part of their Flag Embedding project
- State-of-the-art performance on relevant benchmarks
- Available on HuggingFace

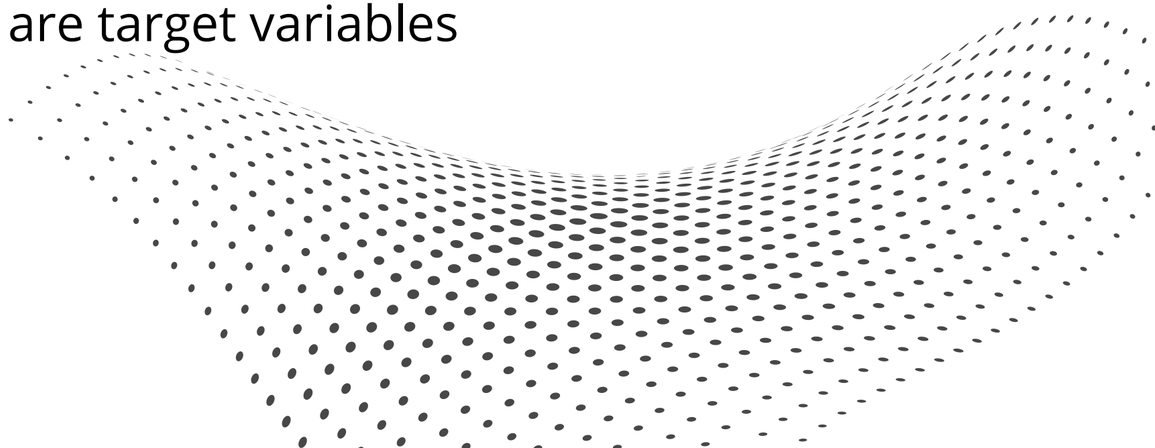
Random forest classifier

Model information:

- In parallel trains a series of classification trees
- Trees are trained on random subsets of data to prevent overfitting
- Trees’ predictions are averaged to give final results

Multi-output case:

- One-vs-rest strategy for training a classifier for each target output
- Requires training as many models as there are target variables



	precision	recall	f1-score	support
Attraction-Inform	1.00	0.99	1.00	994
Attraction-Request	1.00	0.97	0.98	682
Hospital-Inform	1.00	1.00	1.00	3
Hospital-Request	1.00	0.75	0.86	4
Hotel-Inform	1.00	0.99	0.99	3824
Hotel-Request	0.98	0.96	0.97	707
Police-Inform	1.00	1.00	1.00	4
Police-Request	1.00	1.00	1.00	1
Restaurant-Inform	1.00	0.99	0.99	3649
Restaurant-Request	0.99	0.97	0.98	940
Taxi-Inform	1.00	0.99	1.00	717
Taxi-Request	1.00	0.98	0.99	118
Train-Inform	1.00	0.99	1.00	1784
Train-Request	0.99	0.98	0.99	471
general-bye	1.00	1.00	1.00	683
general-greet	1.00	1.00	1.00	24
general-thank	1.00	1.00	1.00	1705
micro avg	1.00	0.99	0.99	16310
macro avg	1.00	0.98	0.98	16310
weighted avg	1.00	0.99	0.99	16310
samples avg	0.99	0.99	0.99	16310

Labeling errors

E.g.

- Utterance: That is everything I need.
- Ground Truth: ['Attraction-Request']
- Prediction: ['']

- Utterance: Yes, please! Ground Truth: ['Restaurant-Request']
- Utterance: Yes please. Ground Truth: ['Hotel-Inform']

True errors

Caused by no context

- Utterance: Can you please provide me with the address, postcode and phone number?
- Ground Truth: ['Hotel-Request']
- Prediction: ['']

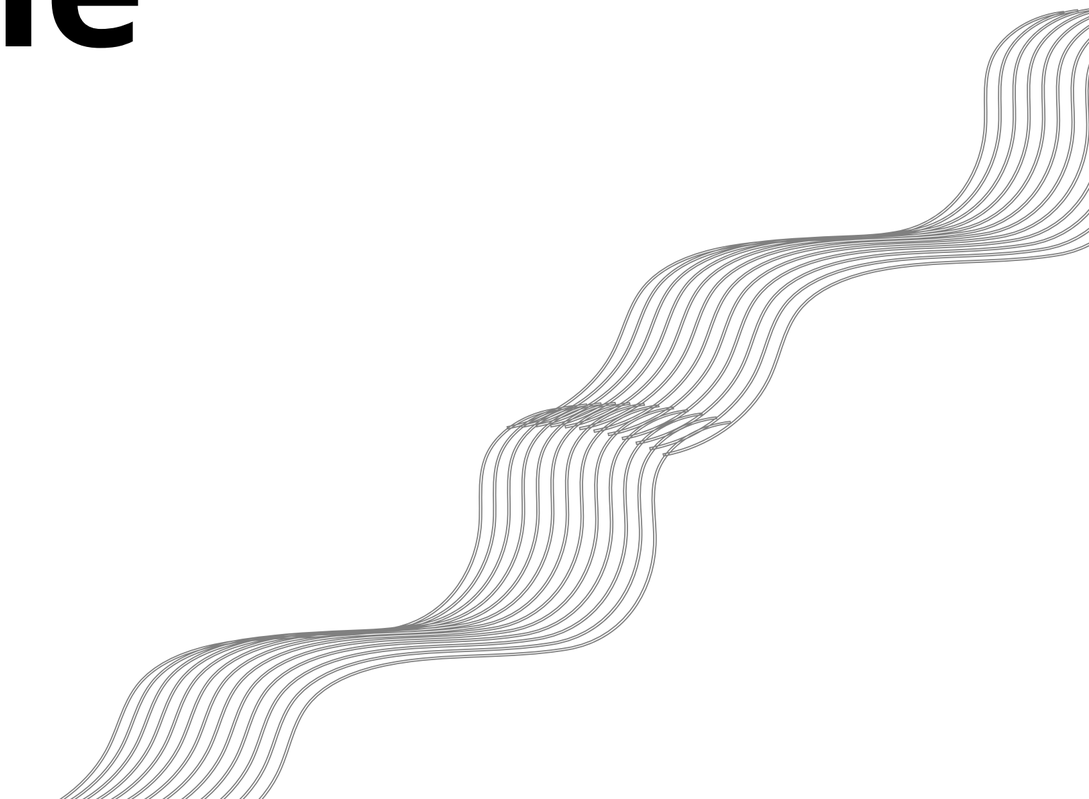
Inexplicable

- Utterance: Does it have wifi?
- Ground Truth: ['Hotel-Request']
- Prediction: ['Hotel-Inform']

Possible improvements

1. Cleaning training and testing data
2. Optimize hyperparameters
3. Account for more context

Semantic frame slot filling



Main idea

Train a Named Entity Recognition (NER) system for detecting slot entities

Two possible approaches:

- **Implement a rule-based system:**
  - Hard to implement
  - May not work well with non-categorical slots (*name*...) or non easily identifiable ones (affirmations, negations, questions, *dontcares*...)

- **Implement a ML-based system:**
  - Train a model from scratch (very costly)
  - Employ transfer learning

Initial idea

We will also need a way to identify affirmations, negations, questions and indifference

Mixed approach:

- **Treat direct mentions as entities:**
  - User: I don’t care about **parking** but I need **wifi**
    - **parking:** *‘parking_dontcare‘*
    - **wifi:** *‘wifi_yes‘*
  - User: Could I get the **phone number**? Also, **any time** is OK
    - **phone number**: *‘phone_question’*
    - **any time**: *‘booktime_dontcare’*

Initial idea

We will also need a way to identify affirmations, negations, questions and indifference

Mixed approach:

- **Identify indirect mentions via sentence classification and dialogue history:**
  - User: I don’t mind
    - Sentence classified as *‘dontcare’*
    - Last information requested to the user: [**food**: *‘?’*]
      - **food**: *‘dontcare’*

NER models

We have explored two different libraries:

- **SpaCy**
  - General purpose NLP library
  - Offers NER as part of its pipeline
  - Allows training and fine-tuning models
  - The default models are opaque and hard to find in the documentation
- **Flair**
  - Lower level NLP library focused on sequence tagging and text classification
  - Allows declaring models programatically
  - Allows inspecting the underlying classifiers

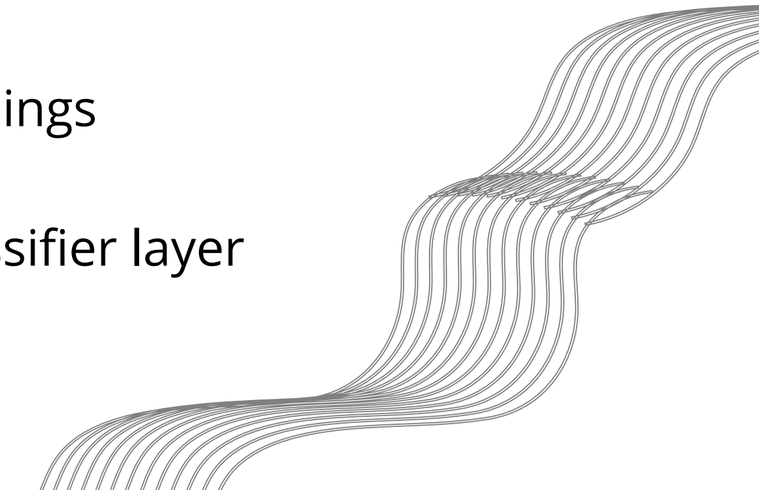
Both of them had very similar performance, so we chose Flair due to its interpretability

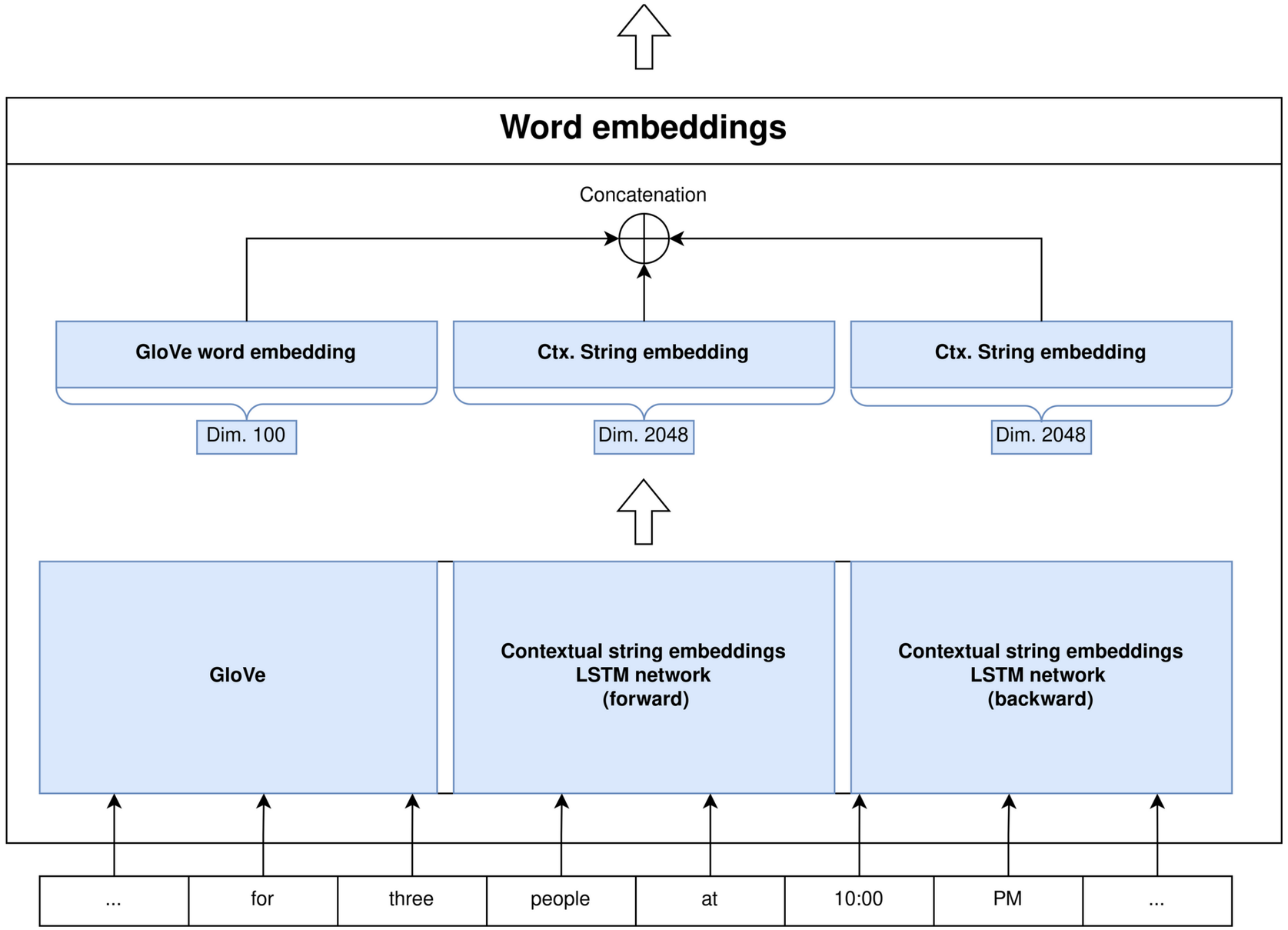
NER models

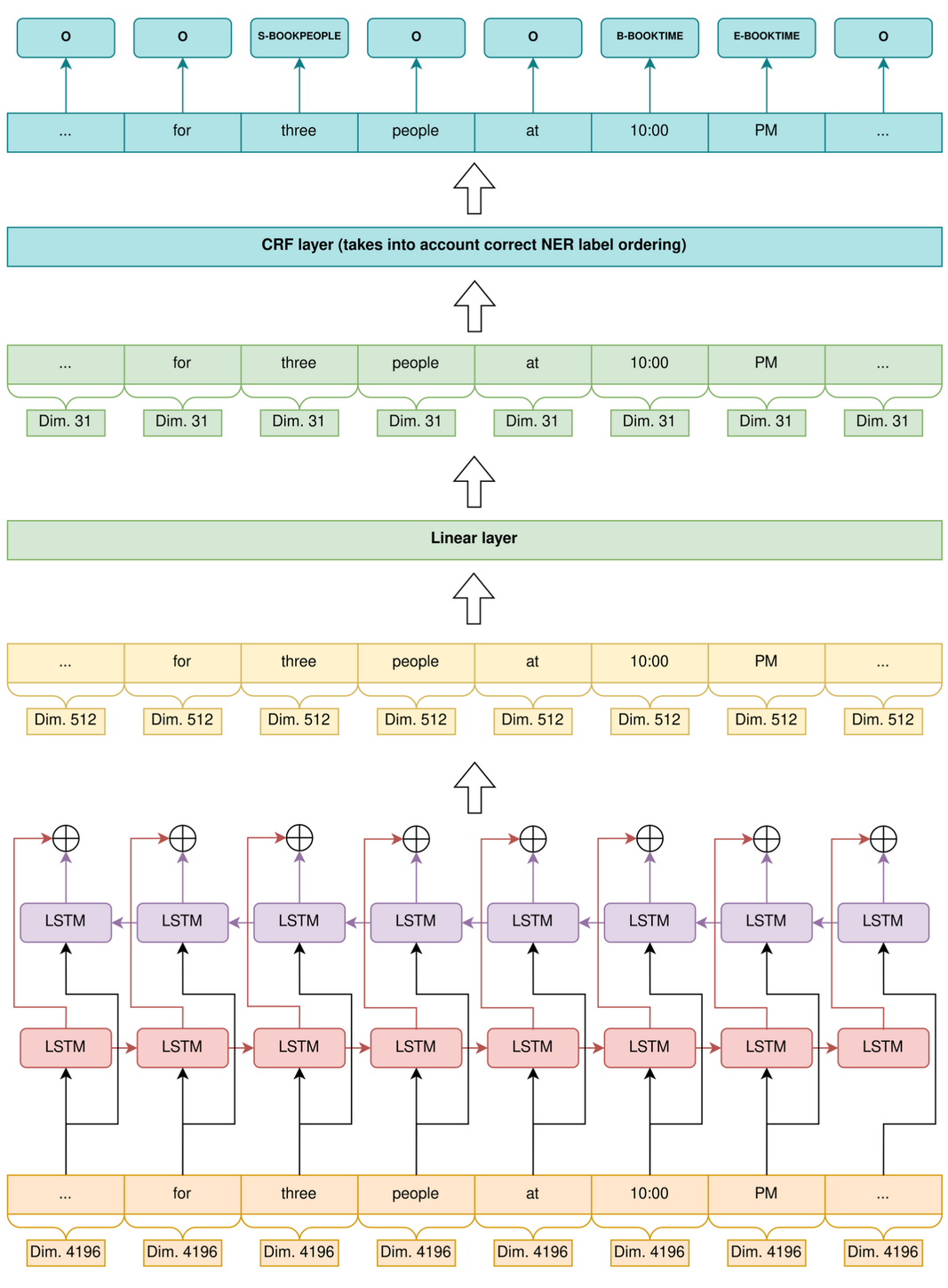
Flair allows easily modifying the embeddings and classifier architectures

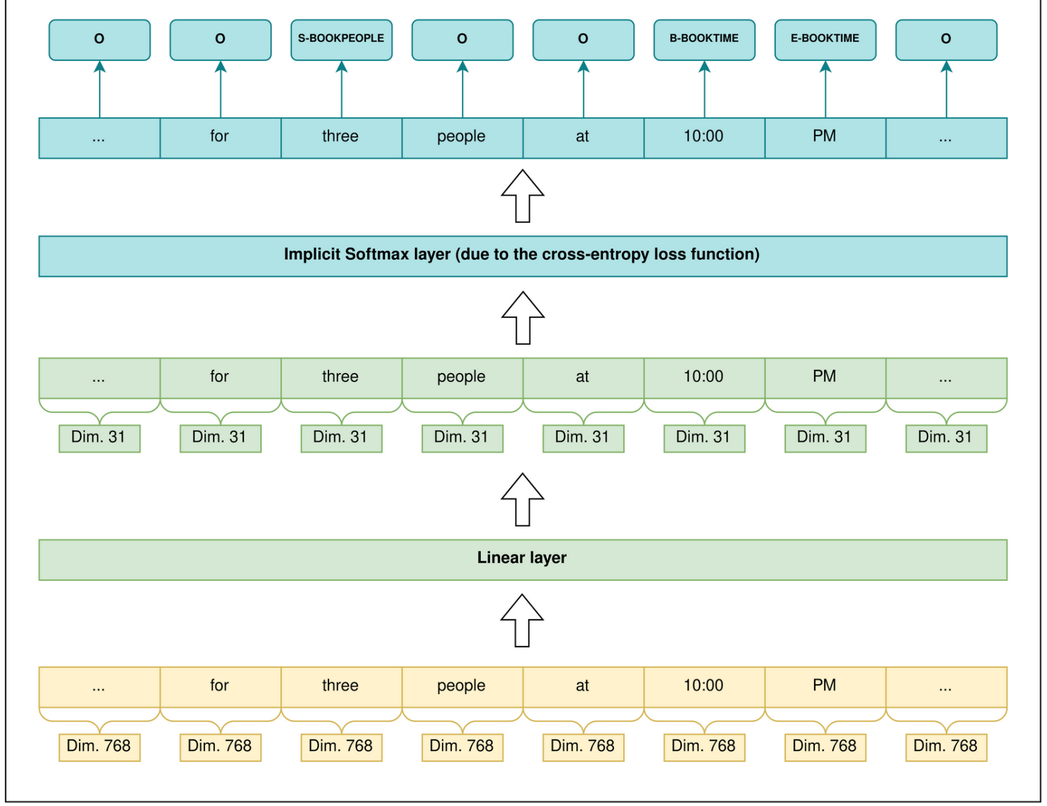
- **Static embeddings NER model:**
  - **Input:** GloVe embeddings concatenated to character-level contextual embeddings
  - **Classifier:** Bidirectional LSTM + Linear layer + CRF
  - **Task:** Train the classifier layer

- **Contextual embeddings model:**
  - **Input:** BERT (DistilBERT, RoBERTa, DeBERTa...) token embeddings
  - **Classifier:** Linear layer + Softmax
  - **Task:** Fine-tune the underlying BERT model and train the classifier layer

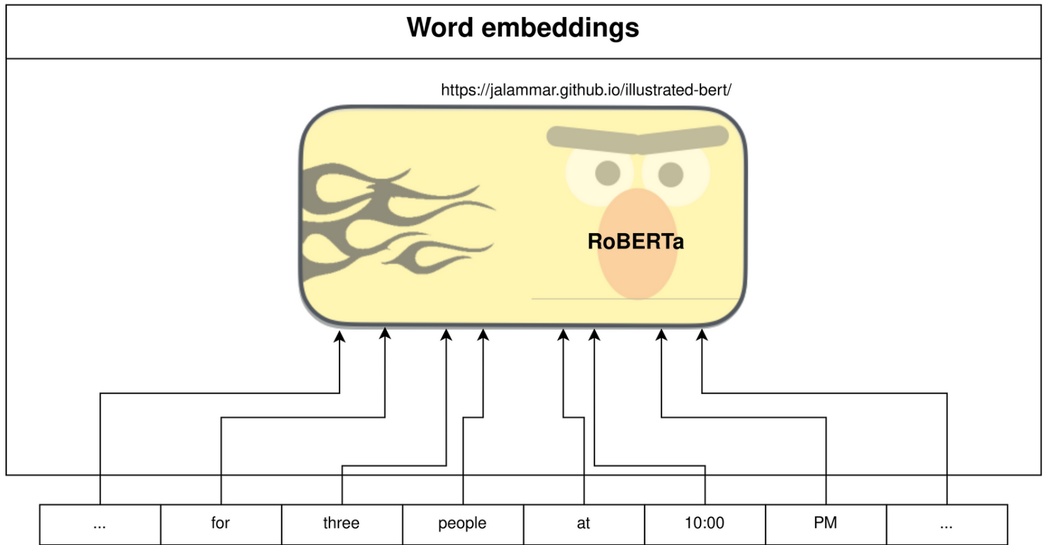



Word embeddings
Concatenation
GloVe word embedding
Ctx. String embedding
Ctx. String embedding
Dim. 100
Dim. 2048
Dim. 2048
GloVe
Contextual string embeddings
LSTM network
(forward)
Contextual string embeddings
LSTM network
(backward)
...
for
three
people
at
10:00
PM
...


O
O
S-BOOKPEOPLE
O
O
B-BOOKTIME
E-BOOKTIME
O
...
for
three
people
at
10:00
PM
...
CRF layer (takes into account correct NER label ordering)
...
for
three
people
at
10:00
PM
...
Dim. 31
Dim. 31
Dim. 31
Dim. 31
Dim. 31
Dim. 31
Dim. 31
Dim. 31
Linear layer
...
for
three
people
at
10:00
PM
...
Dim. 512
Dim. 512
Dim. 512
Dim. 512
Dim. 512
Dim. 512
Dim. 512
Dim. 512
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
...
for
three
people
at
10:00
PM
...
Dim. 4196
Dim. 4196
Dim. 4196
Dim. 4196
Dim. 4196
Dim. 4196
Dim. 4196
Dim. 4196



Extract token embeddings from its last layer



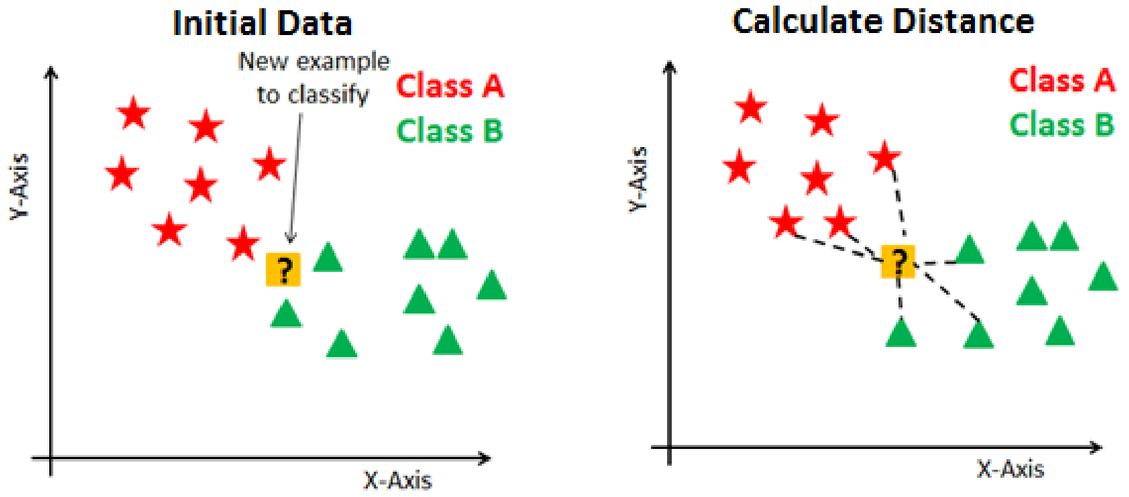
Sentence classification model

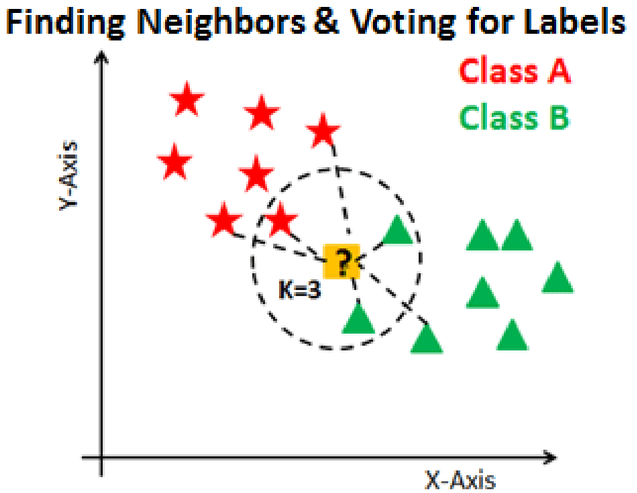
We use the BAAI/bge-large-en-v1.5 SentenceTransformers model:

- We share the model with the User Dialog Act Prediction task

For classification, we use **K-nearest neighbors,** with **no parameter tweaks**

- No downsampling
- Embedding standardization
  - Small improvement in accuracy





Dataset creation

Since the Restaurant and Hotel services only have one conflicting slot name (*'name'*), we trained a single model for both services:

- We inspect the Restaurant and Hotel (slot_name, slot_value) **valid** pairs on every turn
  - If the slot value is ***'dontcare'***, ***'?'***, ***'yes'*** or ***'no'***:
    - Try to find possible terms related to the slot name. If found, create a sample: **(slot_name_{*dontcare*, *question*, *yes*, *no*}, found term)**
  - Otherwise, if the slot value is different and can be found in the utterance, create a sample **(slot_name**, **slot_value)**
    - Exception: If the slot name is *'name'*, we disambiguate it based on the service to avoid collisions
    - If it is not found, try to find related terms

Dataset creation

Other filters and improvements:

- We try match slot values to utterances by using them as-is, in lowercase, in capitalized form and in title form

- Every sample corresponds to a (*utterance*, [*slots*]) pair, with no duplicate utterances with different slots for each one
  - Otherwise, we found that the algorithms were mistakenly learning to only label one or two entities per utterance

- We can combine all shared slots (except *'name'*) across both services
  - This way, we can substantially increase the number of samples

Dataset creation

For the sentence K-nearest neighbors classifiers

- For every utterance with slots in the Restaurant or Hotel domain
  - If it contains only **one slot**
    - And it's a **dontcare**: Add the utterance to the **dontcare** samples
    - And it's a **question**: Add the utterance to the **question** samples
  - Otherwise, add it to the **other** samples

Numbers of samples:

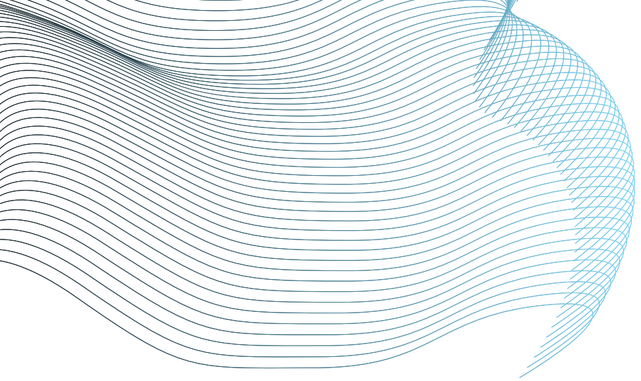
- **dontcare**: 462 samples (train+development), 55 samples (test)
- **question**: 3312 samples (train+development), 361 samples (test)
- **other**: 23533 samples (train+development), 2574 samples (test)

```
train dataset slots:
        Slot area: 4385 items
        Slot pricerange: 4497 items
        Slot food_question: 130 items
        Slot type: 2345 items
        Slot phone_question: 1255 items
        Slot bookday: 4211 items
        Slot bookpeople: 3710 items
        Slot bookstay: 2229 items
        Slot yes_internet: 1263 items
        Slot yes_parking: 1369 items
        Slot pricerange_dontcare: 178 items
        Slot hotel_name: 865 items
        Slot postcode_question: 1035 items
        Slot restaurant_name: 964 items
        Slot booktime: 2497 items
        Slot food: 3775 items
        Slot ref_question: 792 items
        Slot address_question: 1349 items
        Slot stars: 2166 items
        Slot area_question: 265 items
        Slot pricerange_question: 337 items
        Slot parking_question: 145 items
        Slot parking_dontcare: 94 items
        Slot internet_question: 166 items
        Slot food_dontcare: 69 items
        Slot type_question: 90 items
        Slot stars_question: 45 items
        Slot area_dontcare: 310 items
        Slot internet_dontcare: 69 items
        Slot stars_dontcare: 31 items
        Slot no_internet: 61 items
        Slot no_parking: 96 items
        Slot bookday_dontcare: 4 items
        Slot type_dontcare: 8 items
```

```
validation dataset slots:
        Slot area: 510 items
        Slot food: 399 items
        Slot address_question: 133 items
        Slot postcode_question: 139 items
        Slot pricerange_question: 42 items
        Slot yes_parking: 160 items
        Slot stars: 241 items
        Slot type: 271 items
        Slot bookday: 543 items
        Slot bookpeople: 467 items
        Slot bookstay: 282 items
        Slot pricerange: 576 items
        Slot booktime: 347 items
        Slot phone_question: 150 items
        Slot yes_internet: 165 items
        Slot restaurant_name: 144 items
        Slot area_question: 36 items
        Slot stars_question: 12 items
        Slot food_dontcare: 8 items
        Slot hotel_name: 139 items
        Slot food_question: 17 items
        Slot ref_question: 118 items
        Slot type_dontcare: 2 items
        Slot parking_question: 34 items
        Slot pricerange_dontcare: 16 items
        Slot area_dontcare: 26 items
        Slot internet_question: 23 items
        Slot parking_dontcare: 5 items
        Slot invalid: 1 items
        Slot type_question: 17 items
        Slot internet_dontcare: 1 items
        Slot no_parking: 10 items
        Slot no_internet: 3 items
        Slot stars_dontcare: 3 items
```

```
test dataset slots:
        Slot restaurant_name: 150 items
        Slot bookday: 536 items
        Slot booktime: 351 items
        Slot bookpeople: 474 items
        Slot yes_internet: 156 items
        Slot type: 281 items
        Slot yes_parking: 160 items
        Slot pricerange: 545 items
        Slot bookstay: 257 items
        Slot area: 487 items
        Slot food: 407 items
        Slot address_question: 132 items
        Slot postcode_question: 140 items
        Slot stars: 225 items
        Slot pricerange_question: 53 items
        Slot phone_question: 147 items
        Slot parking_question: 25 items
        Slot ref_question: 97 items
        Slot hotel_name: 131 items
        Slot area_dontcare: 35 items
        Slot internet_question: 31 items
        Slot area_question: 41 items
        Slot type_question: 13 items
        Slot pricerange_dontcare: 21 items
```

```
dontcare_possible_slot_values = {
    'pricerange': ['price', 'amount', 'money', 'how much'],

    'area': ['area', 'place', 'where', 'zone', 'wherever', 'location', 'anywhere' ,'town']

    'bookday': ['day', 'when', 'whenever'],

    'food': ['food', 'cuisine', 'style'],

    'booktime': ['anytime', 'any time', 'whenever'],

    'parking': ['parking'],

    'internet': ['internet', 'wifi', 'wi-fi', 'wi fi'],

    'stars': ['stars', 'star', 'rating'],

    'type': ['type', 'kind'],

    'bookstay': ['days', 'time'],

    'ref': ['ref', 'reference number', 'ref number', 'reference'],

    'phone': ['phone number', 'phone'],

    'address': ['address'],

    'postcode': ['postcode', 'post code', 'post', 'code']
}
```



Training

We trained a large number of model configurations:

- One model per service vs. one model for both services
  - No discernible differences, since we already share most slot samples
- Lowercase vs. case-sensitive BERT models
  - No differences, it would depend on the real input (speech-to-text...)
- BERT variants
  - DistilBERT, RoBERTa, DeBERTa-v3, ALBERT
  - All models performed similarly, so we chose DeBERTa-v3 (the most recent one)
- BERT model sizes
  - Base vs. large models
  - The large models did not show any significant improvements and were costlier

Results

Surprisingly, the static embeddings had slighly better results, but the LSTM NNs were too heavy in terms of time

Static embeddings (no extra slots)

```
Results:
- F-score (micro) 0.9637
- F-score (macro) 0.9092
- Accuracy 0.9334
```

DeBERTa-v3 (no extra slots)

```
Results:
- F-score (micro) 0.9592
- F-score (macro) 0.842
- Accuracy 0.9276
```

Static embeddings time cost (no extra slots)

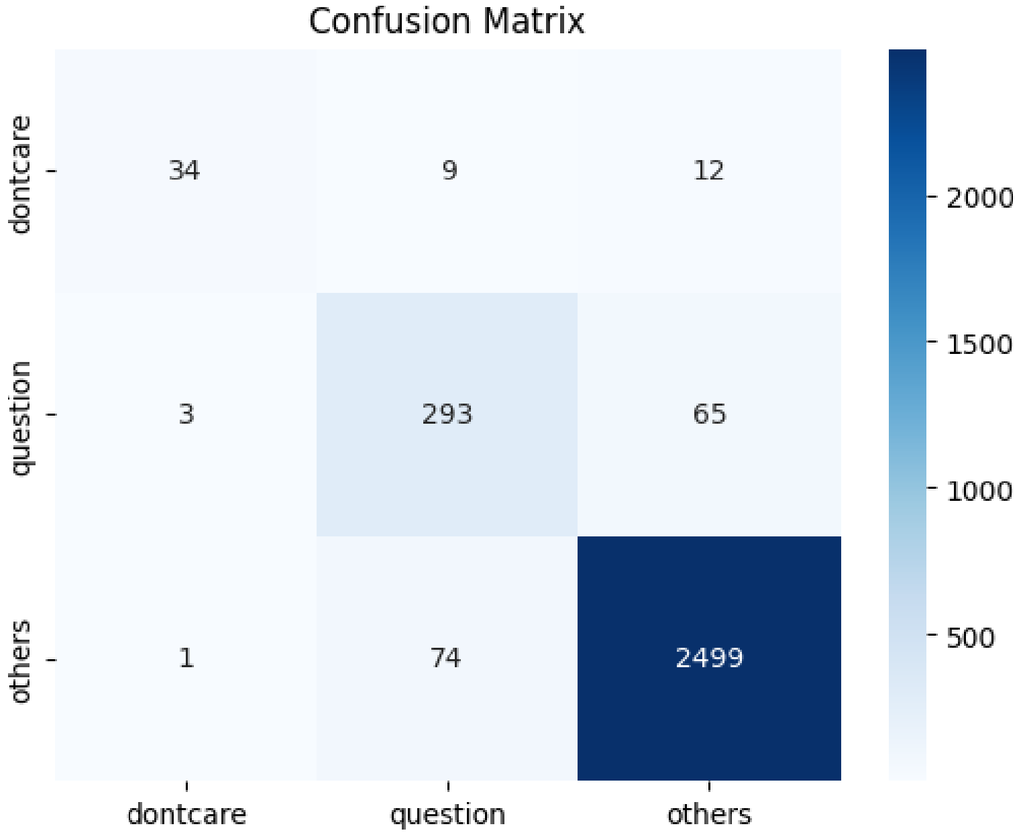
```
Avg. NER time: 0.07346736737447815 s.
Avg. DA pred. time: 2.7242454339907965e-06 s.
Avg. Agent move pred. time: 4.493791945659465e-06 s.
Avg. dialog processing time: 0.5194778014296302 s.
```

DeBERTa-v3 time cost (all extra slots)

```
Avg. NER time: 0.056954742679917 29 s.
Avg. DA pred. time: 2.6587415270251857e-06 s.
Avg. Agent move pred. time: 4.676948778025591e-06 s.
Avg. dialog processing time: 0.40254945180536944 s.
```

Results

The sentence classifier performed very well with no additional tweaks or data cleanup




```
Accuracy (test): 0.95
F1 Score (test, micro): 0.95
F1 Score (test, macro): 0.83
```

	precision	recall	f1-score	support
PRICERANGE	0.9651	0.9633	0.9642	545
BOOKDAY	0.9761	0.9925	0.9843	536
AREA	0.9574	0.9239	0.9403	486
BOOKPEOPLE	0.9466	0.9866	0.9662	449
FOOD	0.9476	0.9755	0.9614	408
BOOKTIME	0.9719	0.9858	0.9788	351
TYPE	0.8196	0.9350	0.8735	277
BOOKSTAY	0.9252	0.9874	0.9553	238
STARS	0.9716	0.9111	0.9404	225
PHONE_QUESTION	0.9290	0.9444	0.9366	180
YES_PARKING	0.7087	1.0000	0.8295	146
RESTAURANT_NAME	0.7287	0.8726	0.7942	157
YES_INTERNET	0.7809	0.9929	0.8742	140
POSTCODE_QUESTION	0.9527	1.0000	0.9758	141
HOTEL_NAME	0.8382	0.8143	0.8261	140
ADDRESS_QUESTION	0.9362	1.0000	0.9670	132
REF_QUESTION	1.0000	1.0000	1.0000	97
PRICERANGE_QUESTION	0.9318	0.8723	0.9011	47
AREA_QUESTION	0.9444	0.8293	0.8831	41
AREA_DONTCARE	0.6905	0.8529	0.7632	34
INTERNET_QUESTION	0.6000	0.3871	0.4706	31
FOOD_QUESTION	0.8421	0.8421	0.8421	19
PRICERANGE_DONTCARE	0.8889	0.3810	0.5333	21
TYPE_QUESTION	0.9286	1.0000	0.9630	13
PARKING_QUESTION	0.0000	0.0000	0.0000	25
FOOD_DONTCARE	0.0000	0.0000	0.0000	10
INTERNET_DONTCARE	1.0000	0.1250	0.2222	8
STARS_DONTCARE	0.0000	0.0000	0.0000	8
STARS_QUESTION	0.0000	0.0000	0.0000	7
NO_PARKING	0.0000	0.0000	0.0000	5
PARKING_DONTCARE	0.0000	0.0000	0.0000	4
INVALID	0.0000	0.0000	0.0000	4
NO_INTERNET	0.0000	0.0000	0.0000	3
TYPE_DONTCARE	0.0000	0.0000	0.0000	1

Final results (our testing dataset)

```
Results:
- F-score (micro) 0.9268
- F-score (macro) 0.6278
- Accuracy 0.8814
```

Final results (evaluation dataset)

Without indirect mentions

```
Extracted information from user's utterance
Precision: 0.791827, Recall: 0.835794, F1-score: 0.813217
```

With indirect mentions (small recall increase)

```
Extracted information from user's utterance
Precision: 0.787987, Recall: 0.839522, F1-score: 0.812939
```

Correct classification examples

User: Sounds good, could I get that **phone number**? Also, could you recommend me an **expensive hotel**?

- **Ground truth**: **pricerange:** *'expensive'*, **'type':** *'hotel'*, **'phone':** *'?'*
- **Predicted:** **pricerange:** *'expensive'*, **'type':** *'hotel'*, **'phone':** *'?'*

User: Guten Tag, I am staying overnight in Cambridge and need a place to sleep. I need free **parking** and **internet**

- **Ground truth:** Nothing
- **Predicted:** **parking:** *'yes'*, **internet**: *'yes'*

User: Any will do, what ever you recommend

- **Ground truth:** **area**: *'dontcare'*
- **Predicted:** **area:** *'dontcare'* (Classified the sentence as *'dontcare'*, and extracted the slots from to-be-requested)

Custom examples

The model is able to detect big numbers of entities, and multiple entities of the same type

I would like a **cheap** **vietnamese**, **french** or **chinese** restaurant for **6** people next **monday** in the city's **east** at **22:30**. Please give me the **phone number**.

Predicted:

- **pricerange**: *'cheap'*
- **food**: *'vietnamese'*, *'french'*, *'chinese'*
- **bookpeople**: *'6'*
- **bookday:** *'monday'*
- **area**: *'east'*
- **booktime**: *'22:30'*
- **phone**: *'?'*

Custom examples

The model is able to detect question/yes/no entities together with others

I want a room for **2** people for **3** nights starting next **tuesday**. I want it to have **parking** and free **wifi**, and i will need its **ref** number"

Predicted:

- **bookpeople**: *'2'*
- **bookstay**: *'3'*
- **bookday:** *'tuesday"*
- **parking**: *'yes'*
- ***internet***: *'yes'*
- **ref**: *'?'*

Custom examples

The model is also able to generalize unseen restaurant and hotel names!

Give me the **price**, **address** and **phone number** for the **krusty krab**, and also for **chum bucket**'s!

Predicted:

- **pricerange**: ‘question’
- **address**: *‘question’*
- **phone:** *‘question‘*
- **restaurant_name**: *‘Krusty Crab’, ‘chum bucket‘*

Misclassification examples

The dataset contains a considerable amount of spurious/arbitrary ground truths

Previous turn: What day are they arriving, and how many nights are they staying?
User: I am unsure at this time.

- **Ground truth**: **name**: *'?'*
- **Prediction**: **bookday**: *'?'*, **bookstay**: *'?'*

Classified the sentence as **'question'**, and extracted the slots from *to-be-requested*

Previous turn: which side of town do you prefer and what is the price range?
User: It doesn't matter. What do you recommend?

- Ground truth: **area**: *'dontcare'*
- Prediction: **area**: *'dontcare'*, **price**: *'dontcare'*

Classified the sentence as **'dontcare'**, and extracted the slots from to-be-requested

Misclassification examples

The model can sometimes miss some entities (but so does the dataset) or part of them

User: Area doesn't really matter, but I would like a **3-star** place with **internet**.

- **Ground truth**: **stars**: *'3'*
- **Prediction**: **internet**: *'yes'*

Ground truth is missing internet
We missed stars

User: I'm planning a trip to Cambridge and I was told the **A and B Guest House** was...

- **Ground truth: name:** *'A and B Guest House'*
- **Prediction: name:** *'Guest House'*

User: i need a place to dine in the **center** thats **expensive**

- **Ground truth: area:** *'centre'*, **pricerange:** *'expensive'*
- **Prediction: area:** *'place'*, **pricerange:** *'expensive'*

Mismatch in ground truth!
It tried to find a related term

Possible improvements

The model does not detect same intents

User: Thank you. Can you also book a hotel room that day for the **same people**?

Idea:

- Add a new **same** slot variant to the NER model, similarly to **dontcare/yes/no/question**
- If the model detects a **{food, bookpeople, area...}_same** entity, look at the existing **Semantic Frames** filled so far, and use the corresponding existing value as the slot value

Possible improvements

The model does not detect both indirect and direct references at the same time

User: **I don’t care**. Could I get the **phone number** please?

- The first sentence refers to the dialogue history, while the second sentence has direct references

Ideas:

- Perform sentence splitting where applicable, and analyze each sentence separately
- Create generic **same**/**dontcare**/**yes**/**no**/**question** entities, to signal the Dialogue Manager to check the dialogue history. Their counterparts that are identifiable as specific slot names would take precedence to avoid collisions

Possible improvements

The model does not care about nonsensical values

User: I want to book a table for **600** people next **thursday** at **29:98**
Predicted:

- **bookpeople**: *‘600’*
- **bookday:** *‘thursday’*
- **booktime**: *‘29:98’*

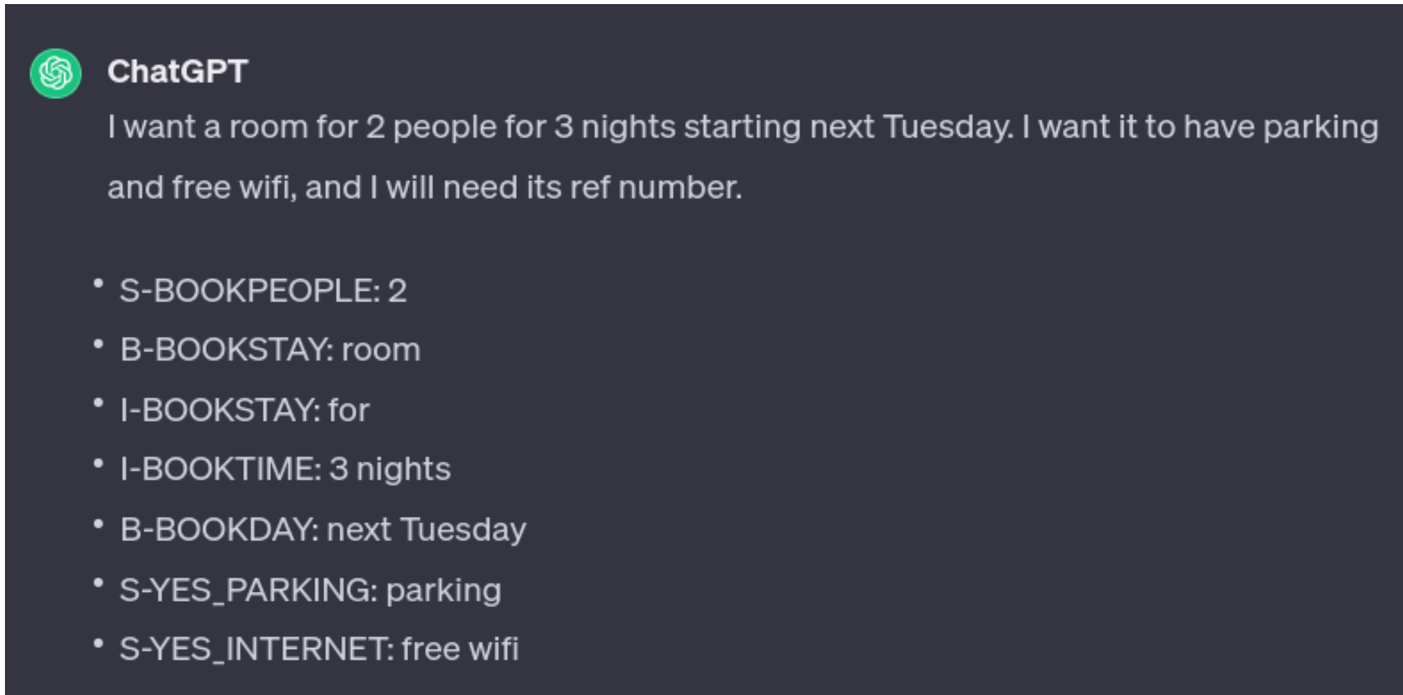
Ideas:

- Sanitize specific values (time, people…)
- Use regexes for categorical values as a first step
  - E.g. for time values: /^([0-9]|0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$/

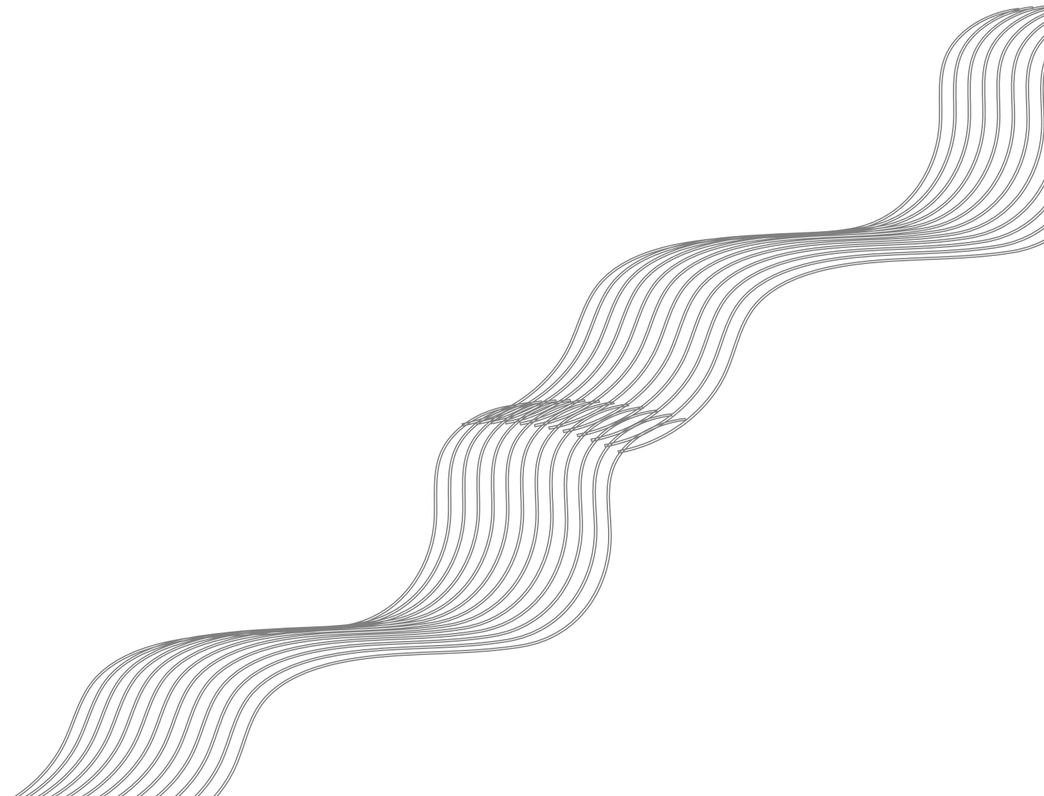
Possible improvements

Other aspects:

- Test large DistilBERT/RoBERTa/DeBERTa/... models
  - Large models were too big to efficiently run in a Colab T4 GPU, and we were hitting the maximum instance time allowed before the training got to finish
- Test encoder-decoder (T5) models
- Test decoder-only (GPT...) models
  - Too costly



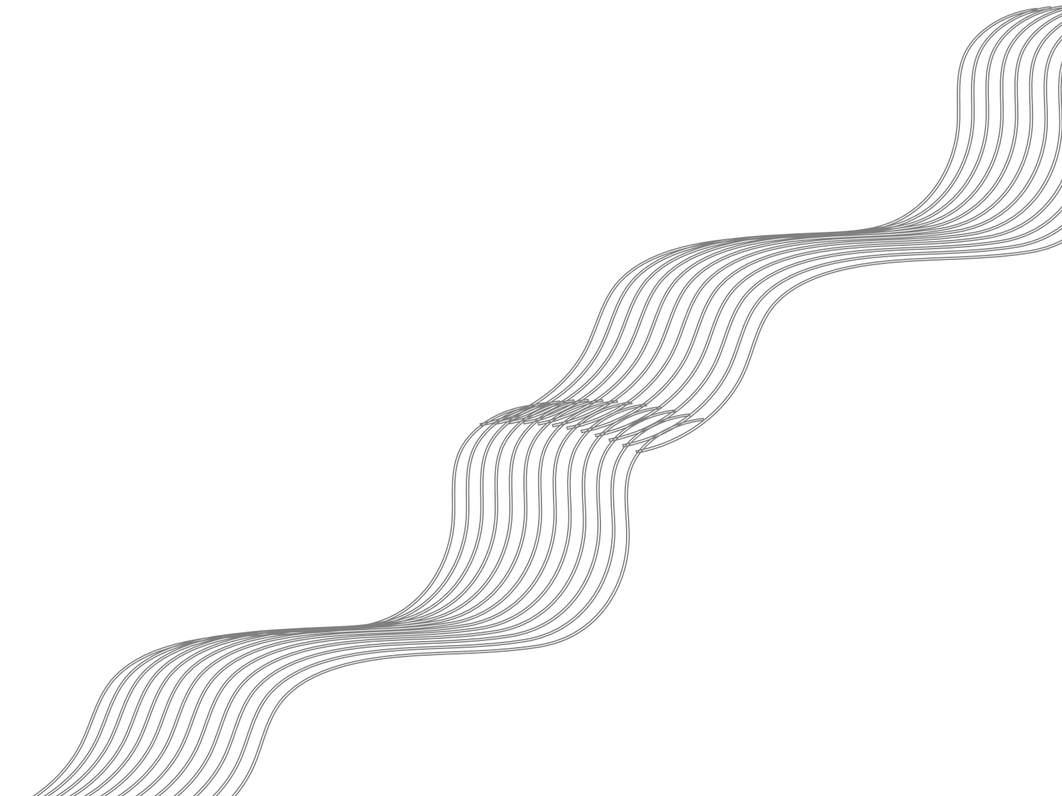
Agent move prediction



Three discrete tasks

1. Agent retrieval prediction
2. Agent dialogue act prediction
3. Agent request prediction

Retrieval prediction



Initial idea

Train a **ML algorithm** for which we need a **multioutput classifier.**

What features could be relevant to determine the output?

- All previous filled slots in the dialogue
- Dialogue acts
- Turn

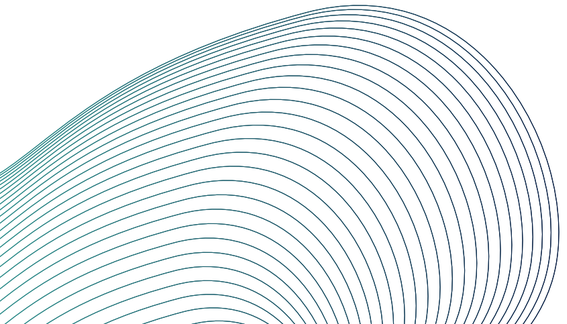
Dataset creation

Since we are restricting to *Restaurant* and *Hotel* domains, we established a list of valid prefixes for each variable taken into account:

- **DA user:** *Restaurant, Hotel, general*
- **Ground Truth:** *restaurant, hotel*

Also we are only targetting certain slots, so we defined a list of valid slots:

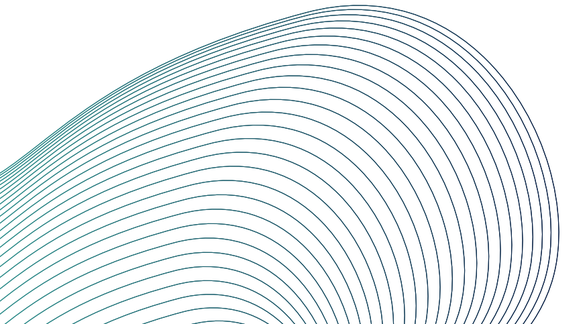
- *pricerange, parking, internet, stars, food, booktime, area, type, bookpeople, bookday, bookstay, name, address, phone, postcode, ref*



Dataset creation

Iterate over all dialogues to get all possible user DA, slots and ground truth values. With this info:

- Create a prototype dictionary for the samples and another for the ground truth.
- For each turn set the ground truth DAs and filled slots to **True** in the sample dictionary and the same for the ground truth *to_be_retrieved* in the ground truth dictionary
- Store each dictionary in a list for each turn
- Create DataFrames with both lists to train the model



restaurant-name: False
hotel-stars: False
restaurant-area: False
hotel-internet: False
hotel-bookday: False
hotel-bookpeople: False
hotel-bookstay: False
restaurant-food: False
hotel-parking: False
hotel-pricerange: False
restaurant-bookday: False
restaurant-pricerange: False
restaurant-booktime: False
restaurant-bookpeople: False
hotel-area: False
hotel-type: False
hotel-name: False
Restaurant-Request: False
Hotel-Request: False
general-thank: False
general-greet: False

Restaurant-Inform: False
Hotel-Inform: False
general-bye: False
Restaurant-Recommend: False
Hotel-Recommend: False
Hotel-Select: False
Booking-Request: False
general-reqmore: False
Hotel-NoOffer: False
Booking-Book: False
Restaurant-Select: False
Restaurant-NoOffer: False
general-welcome: False
Booking-Inform: False
Booking-NoBook: False

hotel-internet: False
restaurant-area: False
restaurant-postcode: False
hotel-parking: False
hotel-type: False
restaurant-booktime: False
hotel-area: False
restaurant-ref: False
hotel-address: False
hotel-booktime: False
restaurant-choice: False
hotel-bookstay: False
restaurant-name: False
restaurant-food: False
restaurant-bookday: False
restaurant-pricerange: False
hotel-name: False
hotel-pricerange: False
hotel-postcode: False
restaurant-bookpeople: False
hotel-phone: False
hotel-bookpeople: False
restaurant-phone: False
hotel-stars: False
hotel-bookday: False
restaurant-address: False
hotel-ref: False
hotel-choice: False

Classifiers

We first tried a few of possible classifiers to see which one was better

Since we are doing multioutput classification we have two main options:

- **Use one that already is multioutput:**
  - Random Forest
- **Use one that is not for multioutput classification and adapt it:**
  - Gradient Boosting
  - Stochastic Gradient Descend
  - Support Vector Machines
  - Logistic Regression

Training

First, dataset restricted to dialogues with services:

- *Restaurant*
- *Hotel*
- *Both*

How to decide which is the best classifier? to keep it simple on training

- **Accuracy:** only takes into account perfect matches
- **Label Ranking Average Precision:** based on ranking probabilities

Training

```
Accuracy: 0.4454545454545544
Label Ranking Average Precision: 0.844238347994599
```

Results for the RF

Results for the other classifiers

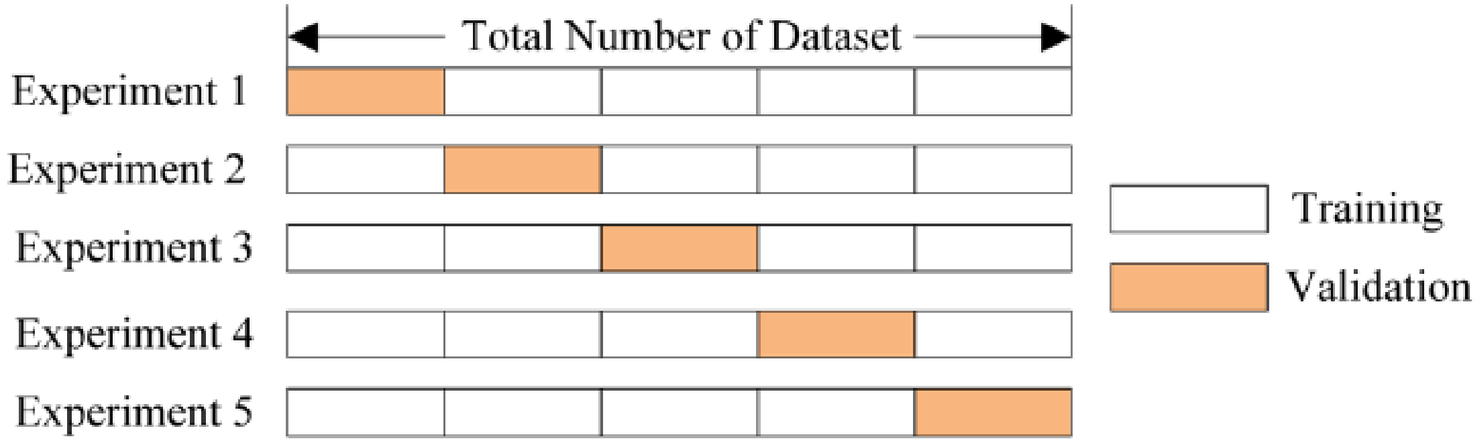
```
GradientBoosting
   - Test Accuracy: 0.4645454545454545
   - Label Ranking Average Precision: 0.8617791617681058
SGDClassifier
   - Test Accuracy: 0.4245454545454546
   - Label Ranking Average Precision: 0.8247449179185213
SVM
   - Test Accuracy: 0.4645454545454545
   - Label Ranking Average Precision: 0.8195261413186447
LogisticRegression
   - Test Accuracy: 0.4454545454545544
   - Label Ranking Average Precision: 0.856115545119652
```

Training

Best ones:

- Random Forest
- Gradient Boosting

Perform k-fold cross validation for the RF in order to find better parameters and retrain both models.



Training

Results for the RF

```
Test accuracy: 0.4818181818181818
Label Ranking Average Precision: 0.8638521589068492
```

This is the one that performs better in terms of F1-score in the evaluation notebook

Results for the GB

```
Test accuracy: 0.4718181818181818
Label Ranking Average Precision: 0.8635490716001871
```

Training

Realized maybe if we were not that strict for the services we could train the classifiers with more data and get a better performance.

Restrict to dialogues that have at least *“Hotel”* or *“Restaurant”* between its services.

Retrain RF and GB on new datasets

Training

Results for the RF

```
Test accuracy: 0.6094182825484764
Label Ranking Average Precision: 0.879715047199657 8
```

This is the one that performs better in terms of F1-score in the evaluation notebook

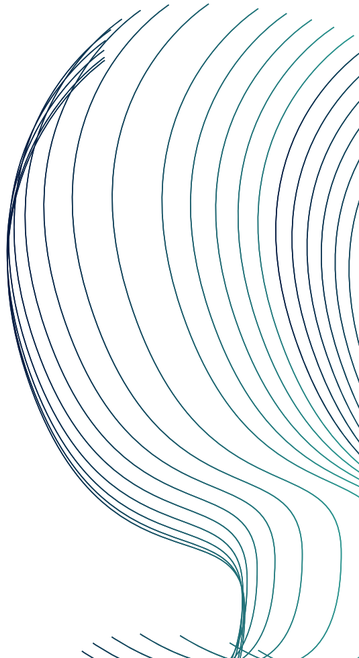
Results for the GB

```
Test accuracy: 0.6248268698060941
Label Ranking Average Precision: 0.8936758835161688
```

Misclassification examples

Little information about filled slots and DAs

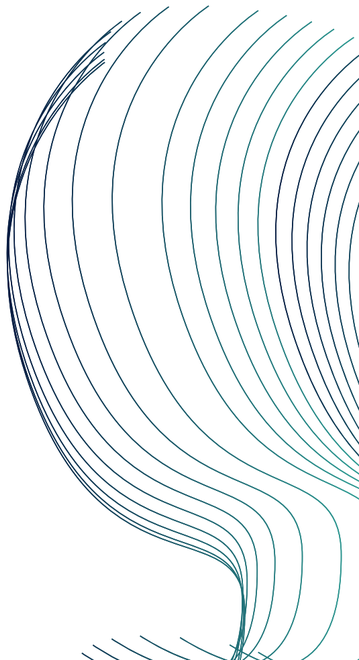
- **Can you find something with italian food instead?**
  - Predicted to be retrieved: ['restaurant-pricerange', 'restaurant-choice']
  - Ground Truth to be retrieved: ['restaurant-name', 'restaurant-food', 'restaurant-choice']
  - Filled slots and dialogue acts: ['restaurant-area', 'restaurant-food', 'Restaurant-Inform']
- **Hello, I am looking for a restaurant in Cambridge. I believe it is called Golden Wok.**
  - Predicted to be retrieved: ['restaurant-address', 'restaurant-name', 'restaurant-area']
  - Ground Truth to be retrieved: ['restaurant-address']
  - Filled slots and dialogue acts: ['restaurant-name', 'Restaurant-Inform']



Misclassification examples

Gets confused with filled slots

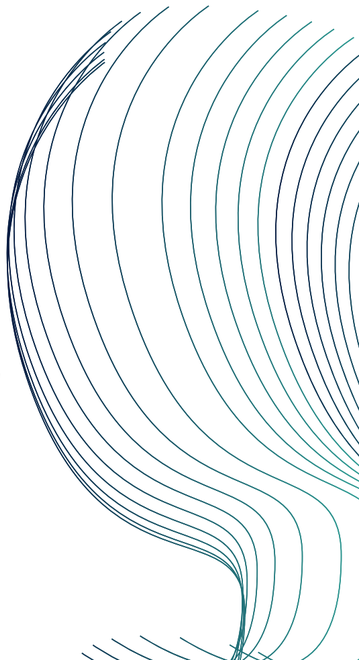
- **A British restaurant, please.**
  - Predicted to be retrieved: ['restaurant-name']
  - Ground Truth to be retrieved: ['restaurant-name', 'restaurant-area', 'restaurant-food', 'restaurant-pricerange']
  - Filled slots and dialogue acts: ['restaurant-area', 'restaurant-food', 'restaurant-pricerange', 'Restaurant-Inform']
  -
- **i am looking for the chiquito restaurant bar**
  - Predicted to be retrieved: ['restaurant-address', 'restaurant-name', 'restaurant-area']
  - Ground Truth to be retrieved: ['restaurant-address', 'restaurant-pricerange']
  - Filled slots and dialogue acts: ['restaurant-name', 'Restaurant-Inform']



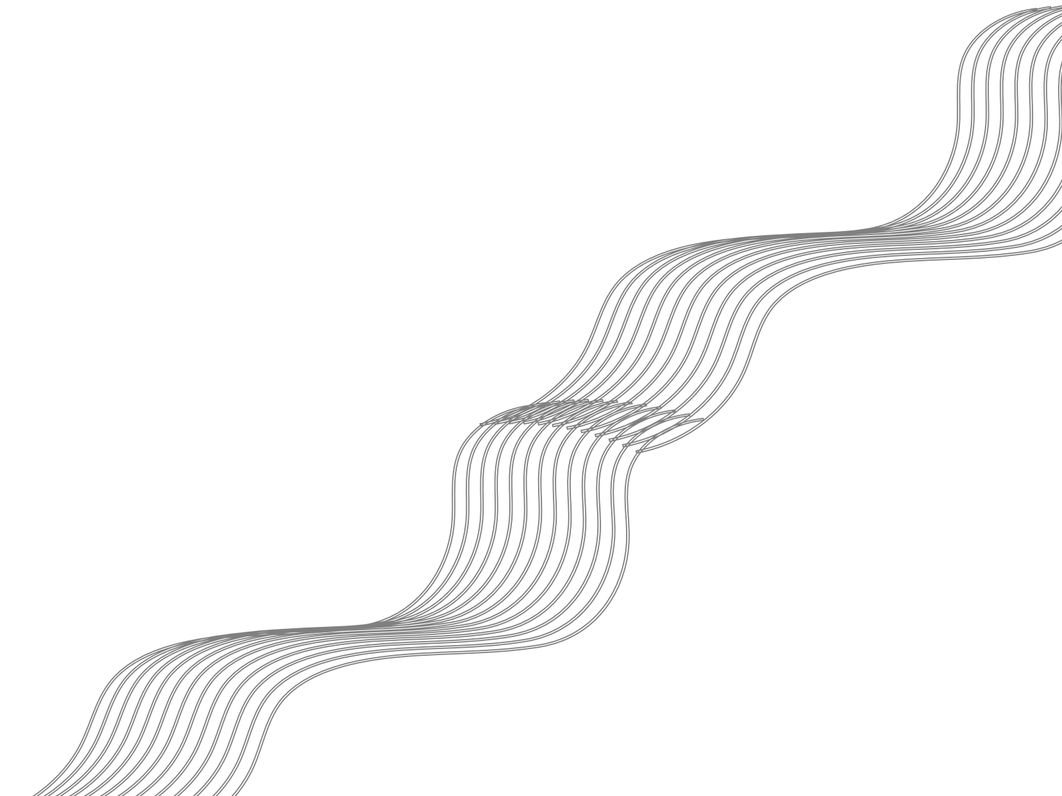
Misclassification examples

"hotel-stars/type" **most of the times is related with the utterance**

- **I just want to confirm that it is actually a hotel, as opposed to a guesthouse?**
  - Predicted to be retrieved: ['hotel-name']
  - Ground Truth to be retrieved: ['hotel-name', 'hotel-type']
  - Filled slots and dialogue acts: ['hotel-name', 'hotel-stars', 'hotel-pricerange', 'hotel-area', 'hotel-parking', 'Hotel-Inform']
- **Does it have a star rating of 2?**
  - Predicted to be retrieved: ['hotel-name']
  - Ground Truth to be retrieved: ['hotel-stars']
  - Filled slots and dialogue acts: ['hotel-stars', 'hotel-pricerange', 'hotel-area', 'hotel-type', 'Hotel-Inform']



Agent act prediction



Problem statement

Predict agent dialogue act (e.g. 'Restaurant-Inform' or 'general-reqmore')

Multi-output binary classification problem

Primary inputs:

- Retrieved information
- Previous user dialogue act
- Previous agent dialogue acts (had no impact on model)
- Turn ID

Predictive model

Classic **structured data prediction** problem

Use XGBoost's multi-output implementation of **Gradient Boosted Classification Trees**

Also tested combining structured data with unstructured data (previous user utterance) and feeding to a **BERT model**; found it performed **poorly**

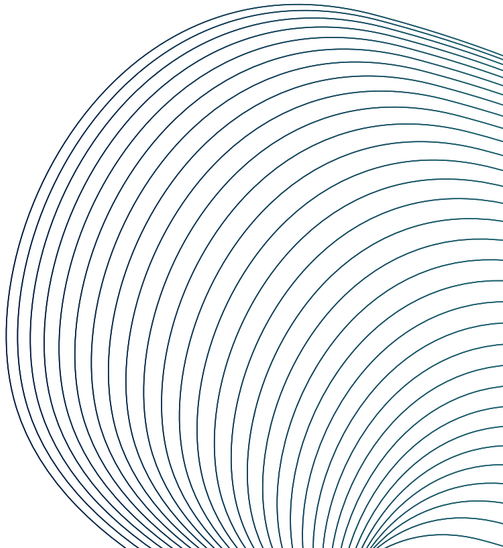
Gradient boosted trees classifier

Model information:

- Sequentially trains a series of classification trees
- Each tree is trained on the misclassified data from an ensemble of previous trees, then added to the ensemble

Multi-output case:

- One-vs-rest strategy for training a classifier for each target output
- Requires training as many models as there are target variables



	precision	recall	f1-score	support
Booking-Book	0.97	0.95	0.96	1364
Booking-Inform	0.45	0.38	0.41	1429
Booking-NoBook	0.74	0.88	0.80	355
Booking-Request	0.42	0.22	0.29	667
Hotel-Inform	0.88	0.91	0.89	2231
Hotel-NoOffer	0.91	0.81	0.86	216
Hotel-Recommend	0.43	0.29	0.34	446
Hotel-Request	0.70	0.41	0.51	880
Hotel-Select	0.24	0.08	0.12	275
Restaurant-Inform	0.91	0.88	0.89	2221
Restaurant-NoOffer	0.94	0.85	0.89	386
Restaurant-Recommend	0.34	0.24	0.28	392
Restaurant-Request	0.66	0.48	0.56	813
Restaurant-Select	0.36	0.18	0.24	250
general-bye	0.00	0.00	0.00	8
general-greet	0.07	0.01	0.01	144
general-reqmore	0.50	0.32	0.39	1740
general-welcome	0.00	0.00	0.00	17
micro avg	0.74	0.60	0.67	13834
macro avg	0.53	0.44	0.47	13834
weighted avg	0.69	0.60	0.64	13834
samples avg	0.66	0.60	0.60	13834

Labeling errors

E.g.

- Utterance: Sure. Does price matter? We can narrow it down and find exactly what you need.
- Ground Truth: ['Hotel-Request', 'Restaurant-Request']
- Prediction: ['']

- Utterance: There are 9 available. What price range do you prefer?
- Ground Truth: ['Restaurant-Inform', 'Restaurant-Request', 'general-greet']
- Prediction: ['Restaurant-Inform', 'Restaurant-Request']

True errors

Lacking context

- Utterance: Sure, when would you like that reservation?
- Ground Truth: ['Booking-Request']
- Prediction: ['']

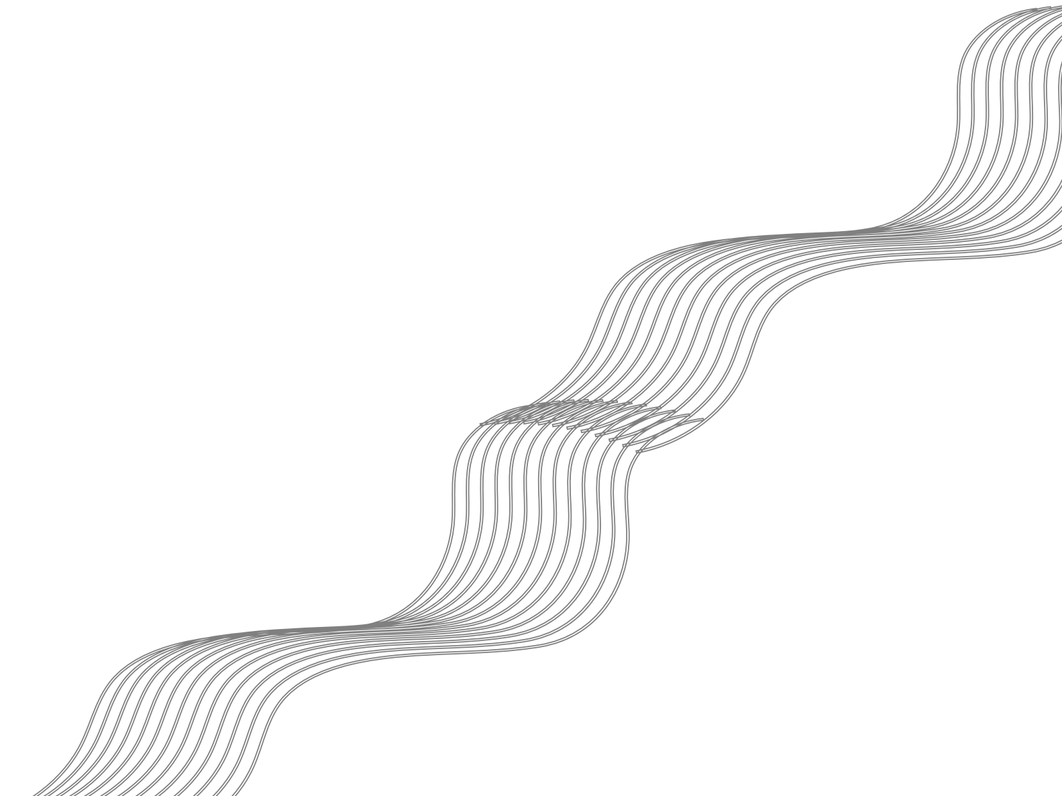
Inexplicable

- Utterance: I have pulled up 3. How about anatolia, they are moderately priced.
- Ground Truth: ['Restaurant-Inform', 'Restaurant-Recommend']
- Prediction: ['Restaurant-Inform']

Possible improvements

1. Cleaning training and testing data
2. Incorporate utterance embeddings
3. Optimize hyperparameters

Requested prediction



Training

Results for the RF

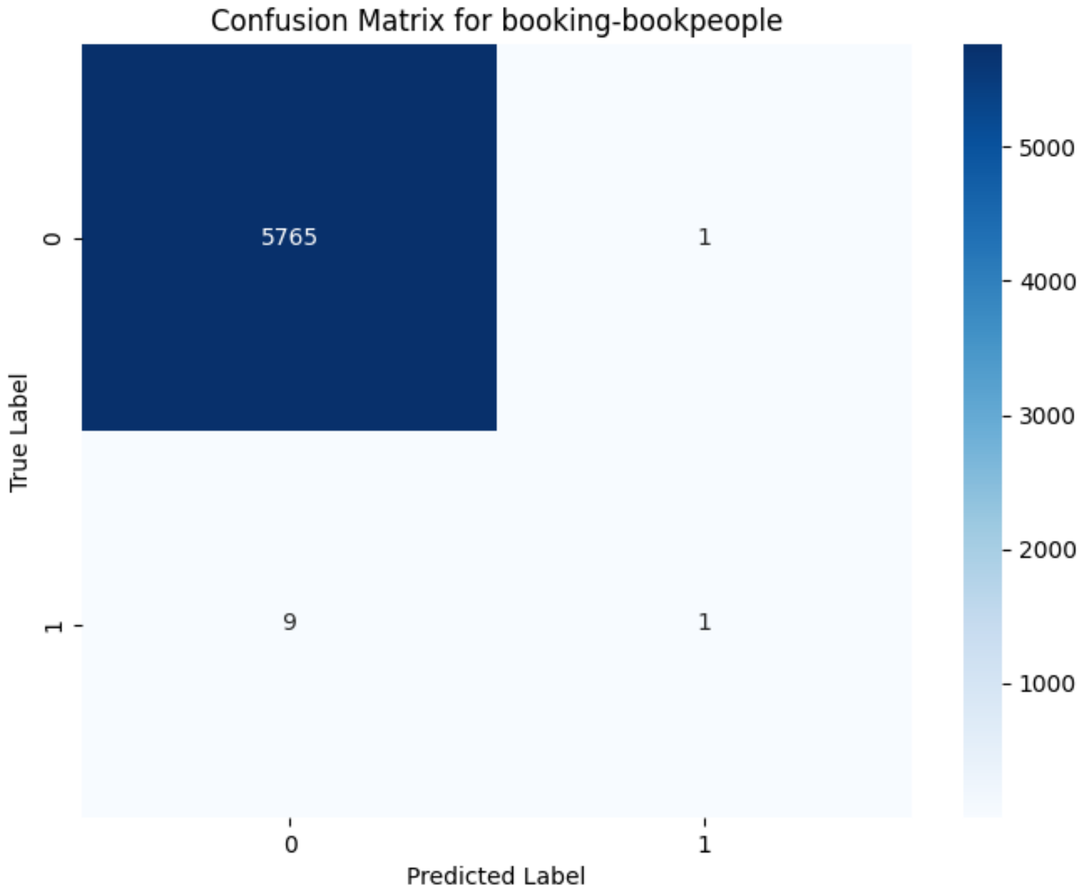
```
Test accuracy: 0.9078947368421053
Label Ranking Average Precision: 0.976730917205293 7
```

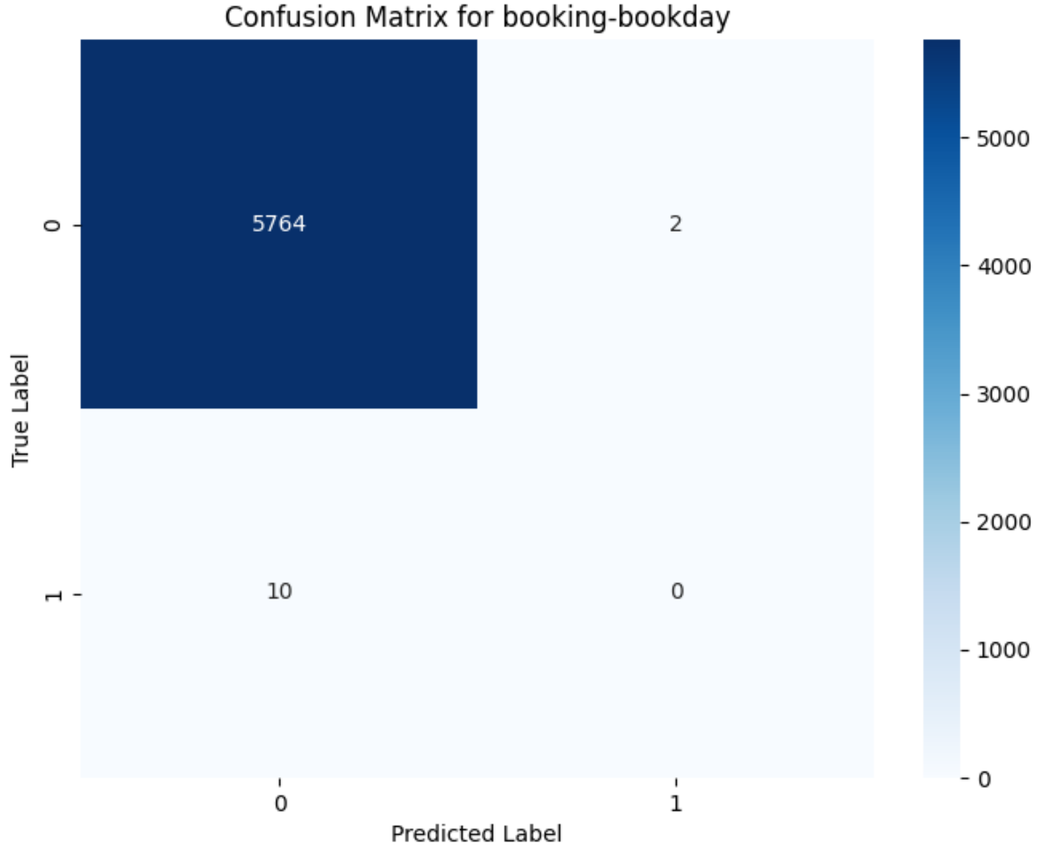
This is the one that performs better in terms of F1-score in the evaluation notebook

Results for the GB

```
Test accuracy: 0.9155124653739612
Label Ranking Average Precision: 0.9776731370641512
```

Misclassification examples





Misclassification examples

Little information, the model is biased towards returning few slots. Average length of ground truth to be requested is 0.217575580830174

- **Can you find me an expensive place serves panasian food?**
  - Predicted to be requested: ['restaurant-food']
  - Ground Truth to be requested: ['restaurant-pricerange']
  - Filled slots and dialogue acts: ['restaurant-food', 'restaurant-pricerange', 'Restaurant-Request', 'Restaurant-Inform', 'Restaurant-NoOffer']
- **I am planning a trip to cambridge, and I am looking for a place to stay. I would like it to include free parking and free wifi a well.**
  - Predicted to be requested: ['hotel-area']
  - Ground Truth to be requested: ['hotel-pricerange', 'hotel-stars']
  - Filled slots and dialogue acts: ['hotel-internet', 'hotel-parking', 'Hotel-Inform', 'Hotel-Request']

Misclassification examples

When there is little info, the model prioritices most common slots in training set.

- “*hotel-area*” appeared 1785 times on training and validation
- *“hotel-pricerange” appeared 1136 times on training and validation*

Previous slide: *“hotel-area”*

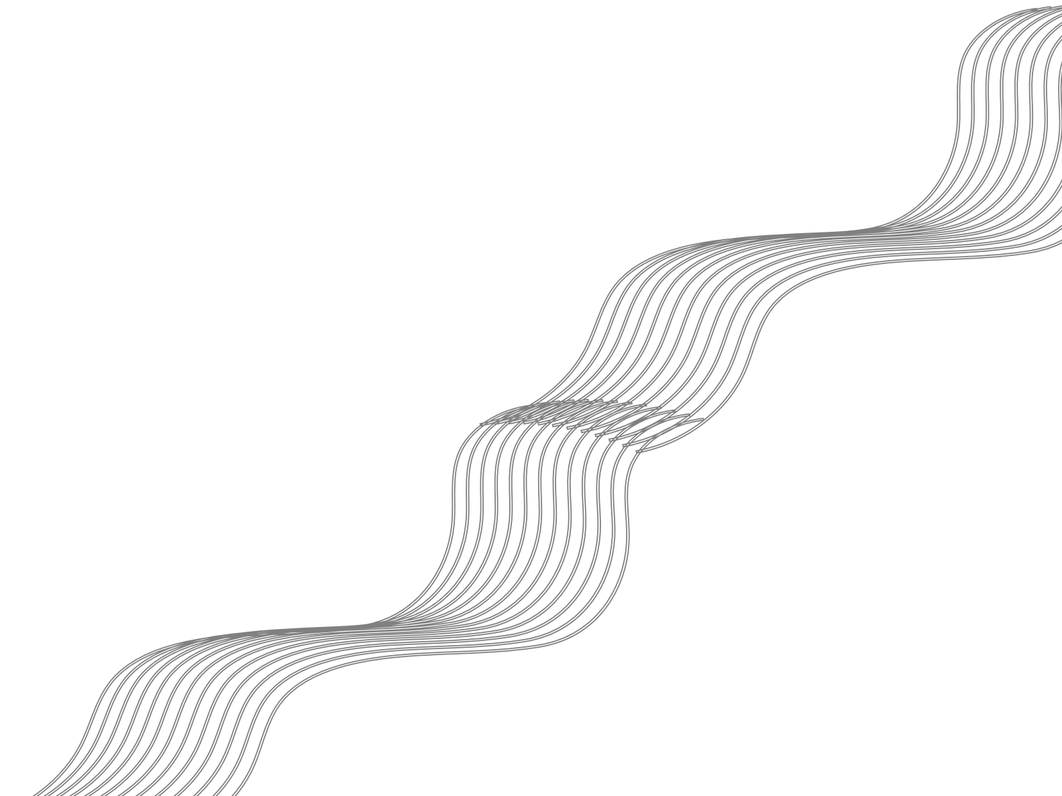
- **north area in a guesthouse**
  - Predicted to be requested: ['hotel-pricerange']
  - Ground Truth to be requested: ['hotel-stars']
  - Filled slots and dialogue acts: ['hotel-type', 'hotel-internet', 'hotel-parking', 'hotel-area', 'Hotel-Inform', 'Hotel-Request', 'turn_number']

Possible improvements

- **Use also the user's utterance**
  - Improve the *"hotel-stars"*, *"hotel-type"*, etc. predictions
  - Improve the predictions when there is little previous information
- **Perform k-fold on the extended dataset**

```
 folds for each of 216 candidates, totalling 1080 fits
END estimator__learning_rate=0.01, estimator__max_depth=None, estimator__min_samples_leaf=1, estimator__min_samples_split=2, estimator__n_estimators=50;, score=0.387 total time= 1.0min
END estimator__learning_rate=0.01, estimator__max_depth=None, estimator__min_samples_leaf=1, estimator__min_samples_split=2, estimator__n_estimators=50;, score=0.382 total time=  58.7s
END estimator__learning_rate=0.01, estimator__max_depth=None, estimator__min_samples_leaf=1, estimator__min_samples_split=2, estimator__n_estimators=50;, score=0.380 total time= 1.0min
END estimator__learning_rate=0.01, estimator__max_depth=None, estimator__min_samples_leaf=1, estimator__min_samples_split=2, estimator__n_estimators=50;, score=0.376 total time=  57.6s
END estimator__learning_rate=0.01, estimator__max_depth=None, estimator__min_samples_leaf=1, estimator__min_samples_split=2, estimator__n_estimators=50;, score=0.374 total time= 1.0min
END estimator__learning_rate=0.01, estimator__max_depth=None, estimator__min_samples_leaf=1, estimator__min_samples_split=2, estimator__n_estimators=100;, score=0.387 total time= 1.9min
```

Dialogue manager

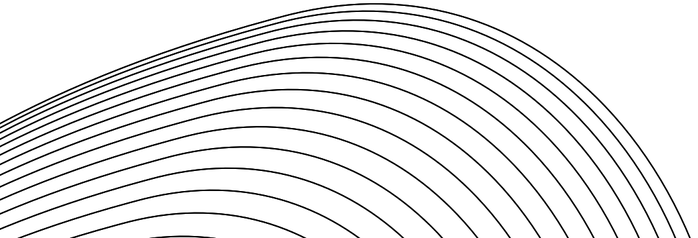


Dialogue manager

Passed relevant information directly between functions to maintain state:

- User utterance embedding
- Turn number
- Relevant ground truth (e.g. previously retrieved information for NER)

Integrated directly into the evaluation loop: lightweight



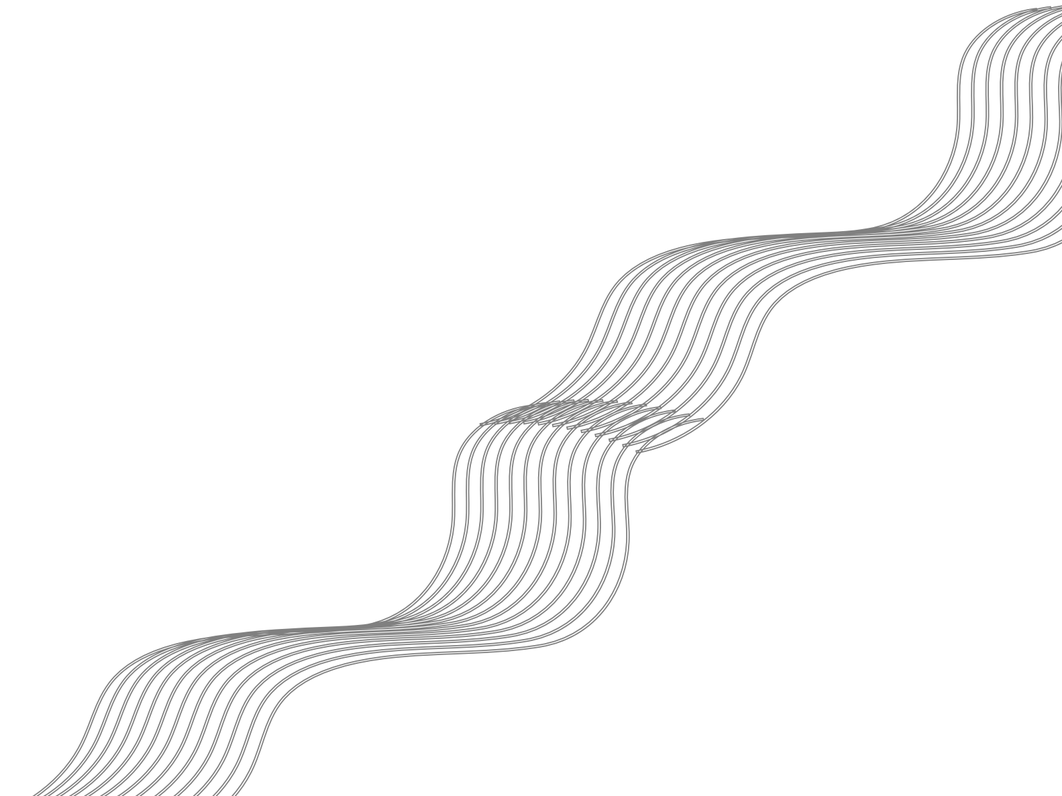
Fast responses (in seconds)

Avg NER time	Avg user DA time	Avg agent move time	Avg utterance embedding time		Avg overall dialogue time
0.04580837936...	**0.019**43824362...	**0.0321**65526725505...	**0.029**291674514472...		**1.088**103259729...

Possible improvements

1. Maintain state to increase context
2. Clean up function structure
3. Generate agent responses

Final results



```
Dialogue 6321/8437:
            Avg. NER time (per turn): 0.045036808589335973 s.
            Avg. DA pred. time (per turn): 0.019079742732236265 s.
            Avg. Agent move pred. time (per turn): 0.0315023341424641 5 s.
            Avg. user utterance embedding pred. time (per turn): 0.028838308633759628 s.
            Avg. dialog processing time: 1.071198073396932 6 s.

            Approx. time remaining: 00:37:46


Dialogue acts in the user's move prediction
Precision: 0.995684, Recall: 0.988897, F1-score: 0.992279

Extracted information from user's utterance
Precision: 0.787987, Recall: 0.839522, F1-score: 0.812939

Info to be retrieved by the agent
Precision: 0.608774, Recall: 0.413239, F1-score: 0.492301

Dialogue acts in the agent's move prediction
Precision: 0.738550, Recall: 0.603099, F1-score: 0.663987

Info to be requested by the agent
Precision: 0.828422, Recall: 0.687616, F1-score: 0.751480
```