Master in Artificial Intelligence

Advanced Human Language Technologies

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Outline

Session 4 - DDI using machine learning

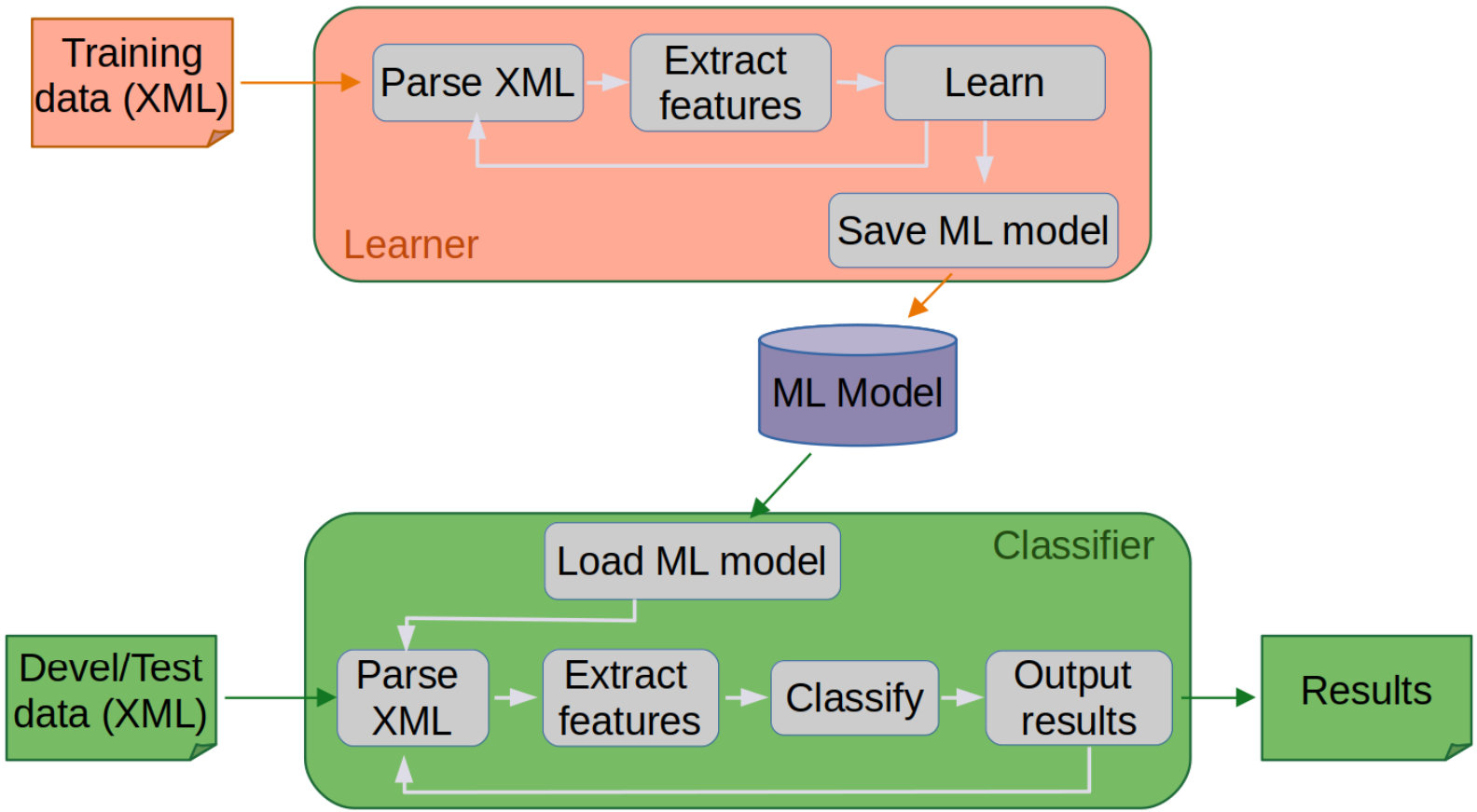
Assignment

Write a python program that parses all XML files in the folder given as argument and classifies drug-drug interactions between pairs of drugs. The program must use a **classification** ML algorithm to solve the problem.

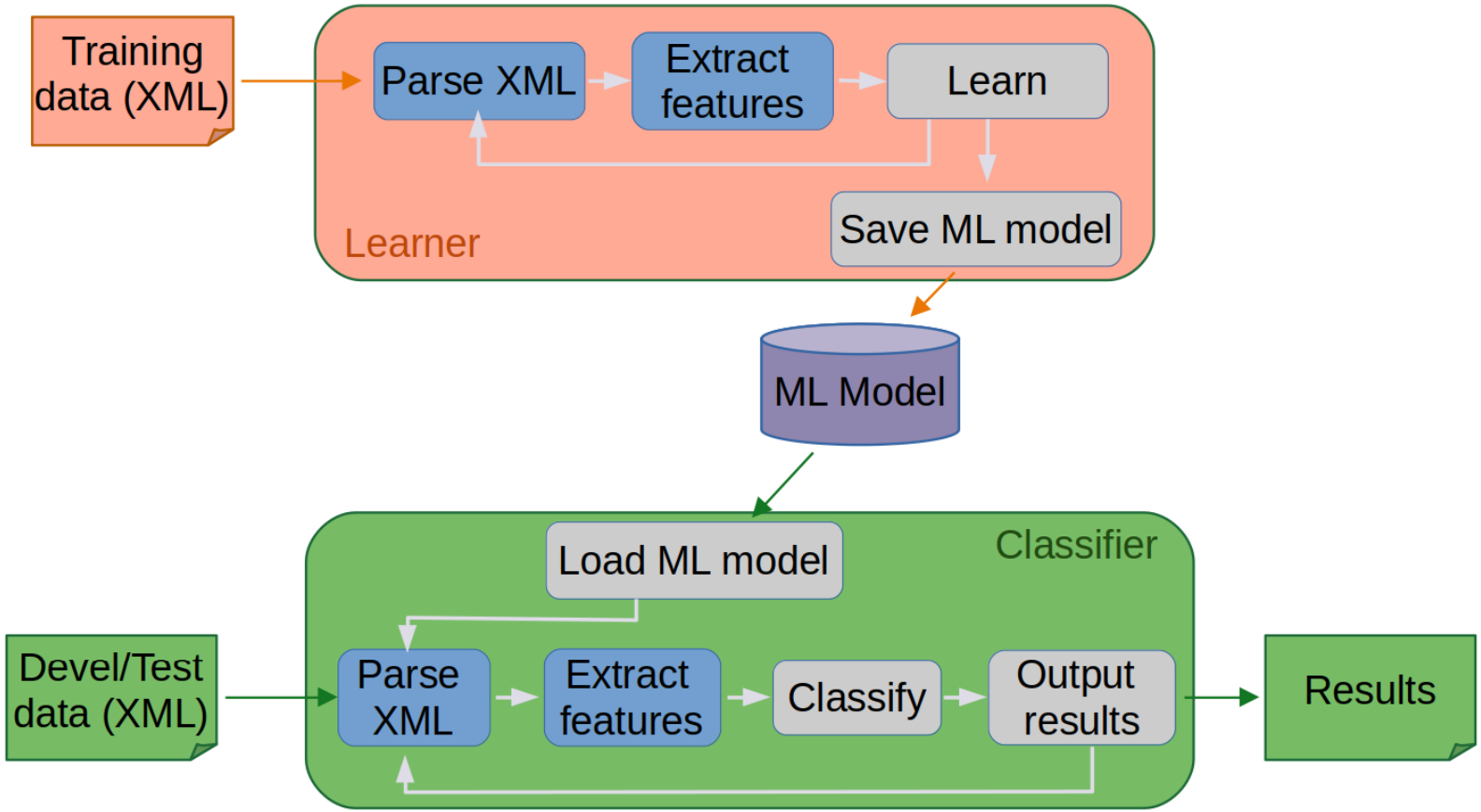
```
$ python3 ./ml-DDI.py data/Devel/
DDI-DrugBank.d278.s0|DDI-DrugBank.d278.s0.e0|DDI-DrugBank.d278.s0.e1|0|null
DDI-MedLine.d88.s0|DDI-MedLine.d88.s0.e0|DDI-MedLine.d88.s0.e1|0|null
DDI-MedLine.d88.s0|DDI-MedLine.d88.s0.e0|DDI-MedLine.d88.s0.e2|0|null
DDI-MedLine.d88.s0|DDI-MedLine.d88.s0.e1|DDI-MedLine.d88.s0.e2|0|null
DDI-DrugBank.d398.s0|DDI-DrugBank.d398.s0.e0|DDI-DrugBank.d398.s0.e1|1|effect
DDI-DrugBank.d398.s0|DDI-DrugBank.d398.s0.e0|DDI-DrugBank.d398.s0.e2|1|effect
DDI-DrugBank.d398.s0|DDI-DrugBank.d398.s0.e2|DDI-DrugBank.d398.s0.e3|0|null
DDI-DrugBank.d398.s1|DDI-DrugBank.d398.s1.e0|DDI-DrugBank.d398.s1.e1|0|null
DDI-DrugBank.d211.s2|DDI-DrugBank.d211.s2.e0|DDI-DrugBank.d211.s2.e5|1|mechanism
DDI-DrugBank.d211.s2|DDI-DrugBank.d211.s2.e1|DDI-DrugBank.d211.s2.e2|0|null
...
```

Outline

General Structure

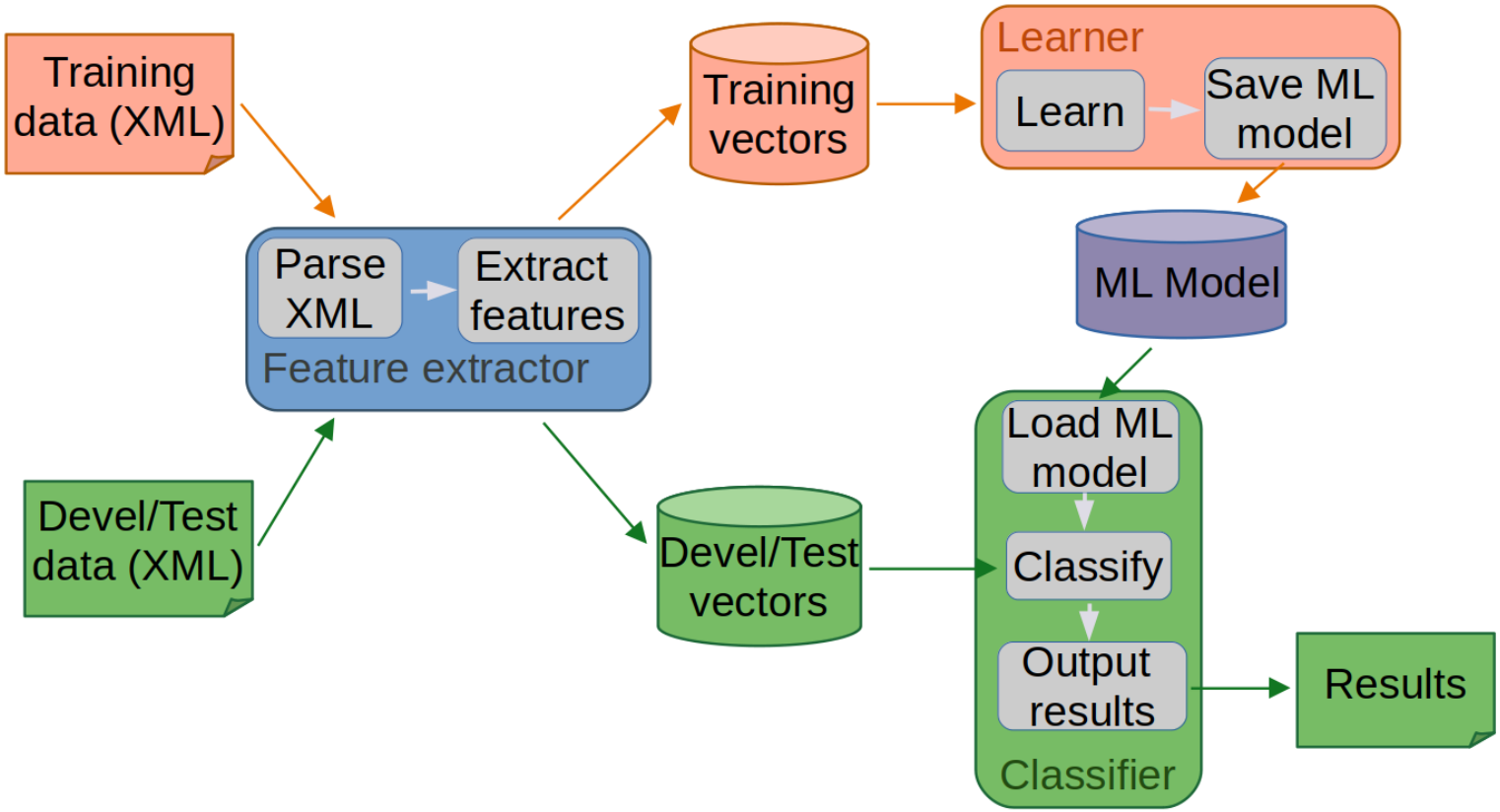


General Structure



Extracting features is a costly operation, which we do not want to repeat for every possible experiment or algorithm parametrization.

General Structure



Feature extraction process is performed once, out of learning or predicting processes.

Outline

Outline

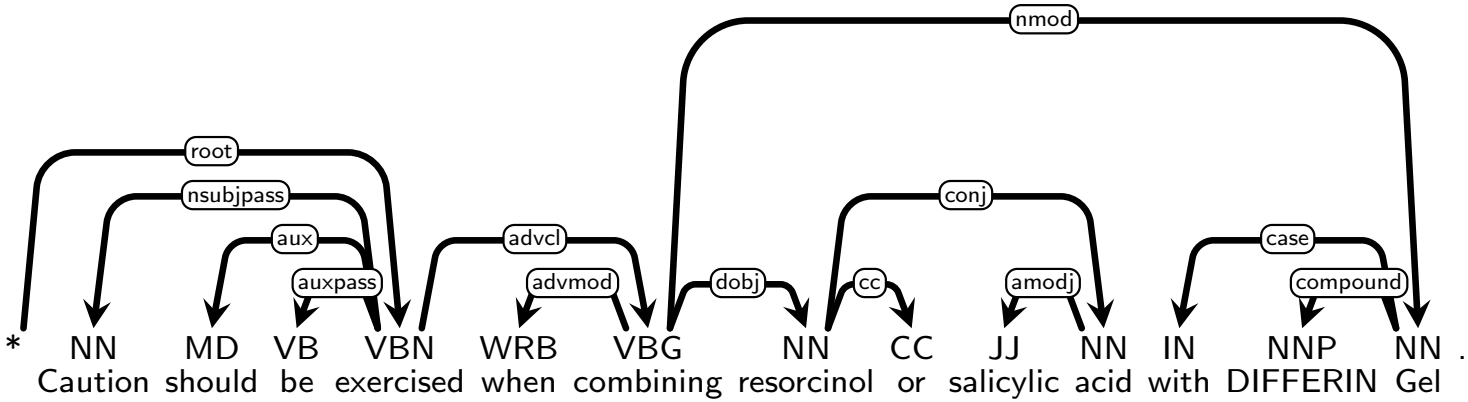
Feature Extractor


```
def analyze(s) :
```

Straightforwardly reuse analyze function from rule-based DDI to call Stanford CORE. Output for the sentence "*Caution should be exercised when combining resorcinol or salicylic acid with DIFFERIN Gel*" should look like:

```
{0:{'head':None,'lemma':None,'rel':None,'tag':'TOP','word':None},
1:{'word':'Caution','head':4,'lemma':'caution','rel':'nsubjpass','tag':'NN','start':0,'end':6},
2:{'word':'should','head':4,'lemma':'should','rel':'aux','tag':'MD','start':8,'end':13},
3:{'word':'be','head':4,'lemma':'be','rel':'auxpass','tag':'VB','start':15,'end':16},
4:{'word':'exercised','head':0,'lemma':'exercise','rel':'ROOT','tag':'VBN','start':18,'end':26}
5:{'word':'when','head':6,'lemma':'when','rel':'advmod','tag':'WRB','start':28,'end':31},
6:{'word':'combining','head':4,'lemma':'combine','rel':'advcl','tag':'VBG','start':33,'end':41}
7:{'word':'resorcinol','head':6,'lemma':'resorcinol','rel':'dobj','tag':'NN','start':43,'end':52
8:{'word':'or','head':7,'lemma':'or','rel':'cc','tag':'CC','start':54,'end':55},
9:{'word':'salicylic','head':10,'lemma':'salicylic','rel':'amod','tag':'JJ','start':57,'end':65}
10:{'word':'acid','head':7,'lemma':'acid','rel':'conj','tag':'NN','start':67,'end':70},
11:{'word':'with','head':13,'lemma':'with','rel':'case','tag':'IN','start':72,'end':75},
12:{'word':'DIFFERIN','head':13,'lemma':'DIFFERIN','rel':'compound','tag':'NNP','start':77,'end'
13:{'word':'Gel','head':6,'lemma':'gel','rel':'nmod','tag':'NN','start':86,'end':88},
```

Feature Extractor



Entities:

e0: `resorcinol` e1: `salicylic acid` e2: `DIFFERIN Gel`

Example path features:

e0 to e1: [e0 $\overset{conj}{\rightarrow}$ e1]. (e1 is direct child of e0. The arc is labeled *conj*).

e0 to e2: [e0 $\overset{dobj}{\leftarrow}$ combine $\overset{nmod}{\rightarrow}$ e2]. (e0 is direct child of verb "combine" with label *dobj*, and e2 is direct child of the same verb, with label *nmod*).

e1 to e2: [e1 $\overset{conj}{\leftarrow}$ resorcinol $\overset{dobj}{\leftarrow}$ combining $\overset{nmod}{\rightarrow}$ e2].

Feature Extractor

`def extract_features(tree, entities, e1, e2) :`

- Input: Receives an analyzed sentence tree, the entities present in the sentence, and the ids of thw two target entities.
- Output: Returns a list of binary features, preceeded by
- Example:

```
>>> extract_features(tree,
{'DDI-DrugBank.d370.s1.e0':['43','52'],
'DDI-DrugBank.d370.s1.e1':['57','70'],
'DDI-DrugBank.d370.s1.e2':['77','88']},
'DDI-DrugBank.d370.s1.e0', 'DDI-DrugBank.d370.s1.e0')

['lb1=Caution', 'lb1=be', 'lb1=exercise', 'lb1=combine',
'lib=or', 'la2=with', 'la2=DIFFERIN', 'la2=Gel', '2under1',
'dep=conj']
```

Feature Extractor

`def output_features(id,e1,e2,type,features) :`

- Input: Receives a sentence id, two entity ids, the gold class, and list of binary features.
- Output: Prints to stdout the feature vector in the following format: one single line per vector, with tab-separated fields: sent_id, ent_id1, ent_id2, gold_class, $feature_1$, $feature_2$, ...

  Note: Field *gold_class* will be used only in training. Fields *sent_id*, *ent_id1*, *ent_id*, will be used in prediction to produce the output format expected by the evaluator (same as the rule-based classifier).
- Example output:

```
DDI-DrugBank.d370.s1 DDI-DrugBank.d370.s1.e0
DDI-DrugBank.d370.s1.e1 null lb1=Caution lb1=be lb1=exercise
lb1=combine lib=or la2=with la2=DIFFERIN la2=Gel 2under1
path=conj
```

Outline

Learner - Option 1: Maximum Entropy

- Use `megam` to train a model as seen in class
- `megam` does not expect the extra information in the features file, so:
  - Remove the first 3 fields (*sent_id*, *ent_id1*, *ent_id*):

    ```
    cat feats.dat | cut -f4- > megam.dat
    ```

  - Alternatively, you can modify the `output_features` function to directly produce two versions of the feature file, one with the extra information, and one without.

Learner - Option 2: Your choice

- Select a ML algorithm of your choice (DT, SVM, RF, ...) and a python library implementing it.
- Adapt the feature file format to the needs of the selected algortihm
- Train a classification model for the task of **classifying** entity pairs.

Note that the target task is a mere classification, not a sequence prediction. So, for a given sentence and a given pair of entities in it, the output is just **one** label, not a sequence. Thus, sequence tagging algorithms such as CRFs are overdimensioned.

Outline

Classifier - Option 1: Maximum Entropy

- Follow examples (and reuse code) for MaxEnt classifiers seen in class to get a label for each vector in the dataset.

Classifier - Option 2: Your choice

- Write the necessary code to call your choice classifier and get a label for each vector in the dataset.

Classifier - Produce output (all options)


```
def output_ddi(id, id_e1, id_e2, outf) :
```

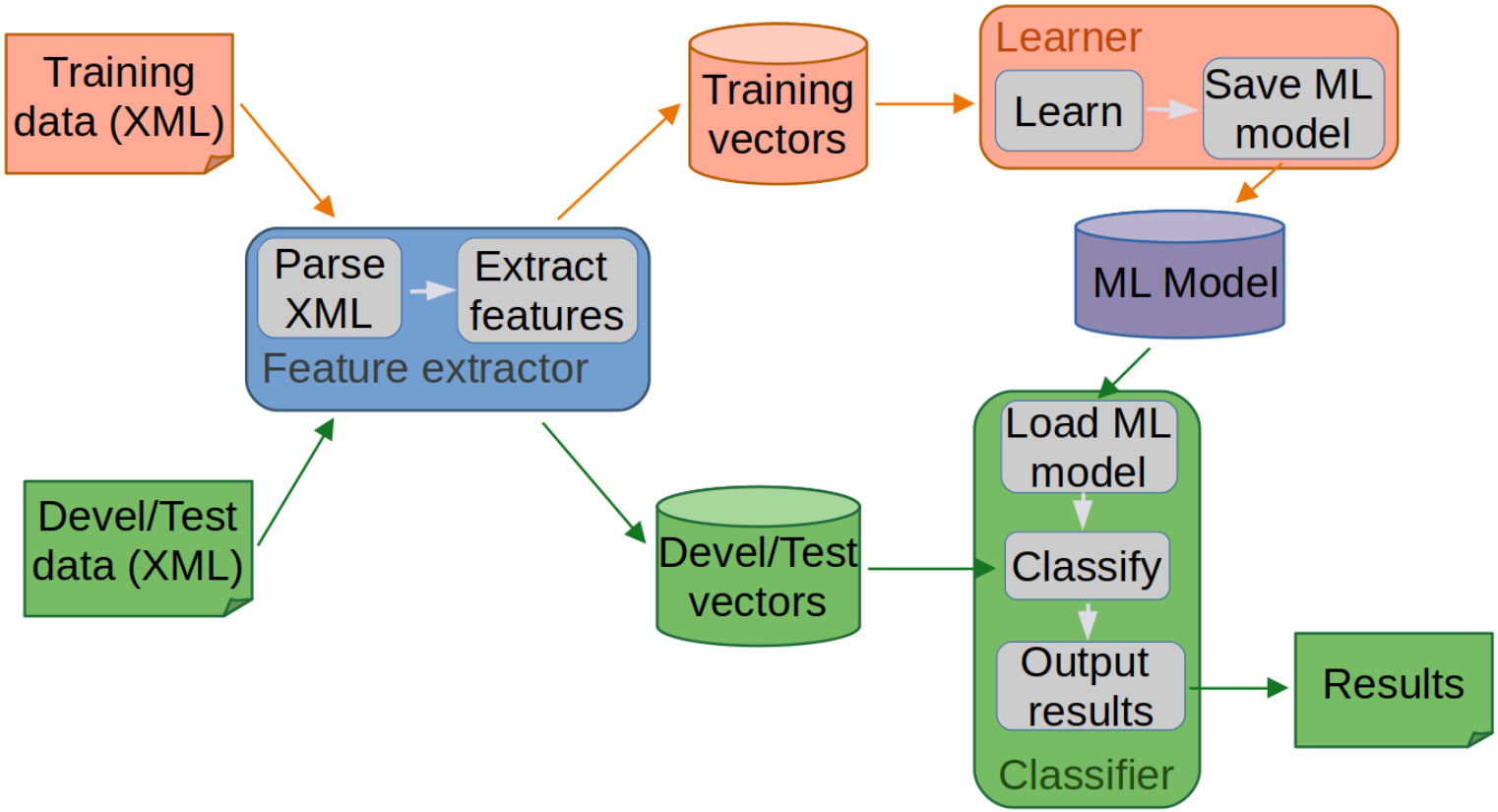
- Input: Receives a sentence id, two entity ids, a predicted class for each token, and an open output file object.
- Output: Prints on `outf` the interactions in the right format: one line per entity, fields separated by '|', field order: *id*, *id_e1*, *id_e2*, *ddi*, *type*. The *ddi* field is redundant: it should be set to 0 when *type*=`null`, and to 1 otherwise.
- Example:

```
>>> output_entities("DDI-DrugBank.d553.s0", "DDI-DrugBank.d553.s0.e1",
"DDI-DrugBank.d553.e2", "effect", sys.stdout)
DDI-DrugBank.d553.s0|DDI-DrugBank.d553.s0.e1|DDI-DrugBank.d553.s0.e2|1|effect
```

Outline

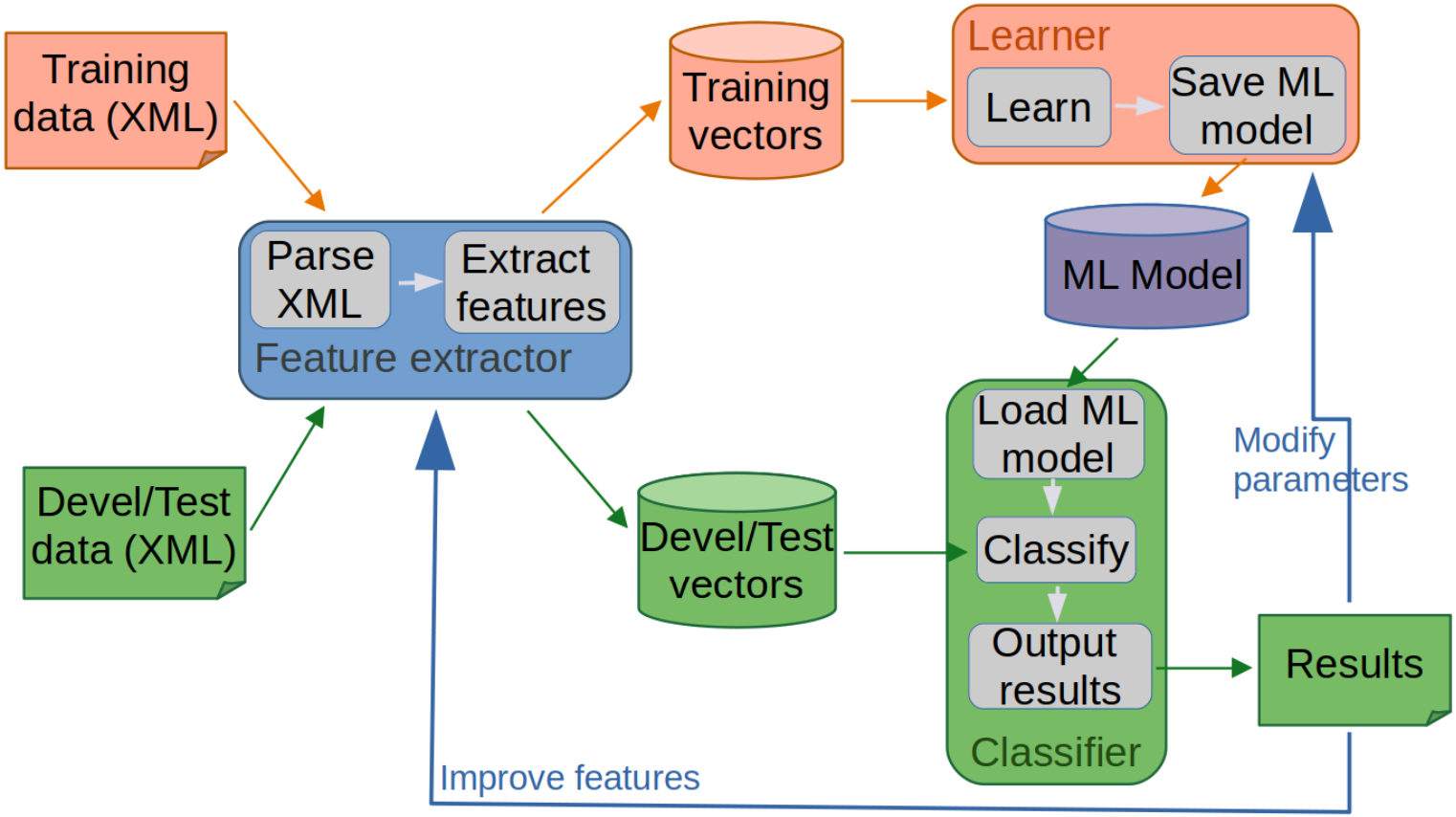
Build a good ML-based DDI detector

Strategy to follow:



Build a good ML-based DDI detector

Strategy to follow:



Outline

Evaluating Results

Use function `evaluate` from previous exercise to obtain performance statistics.

Evaluation goals:

- Find out whether the added feature(s) are useful or damaging
- Find out which is the best parameterization of the algorithm.

Machine Learning Systems Development Methodology

1. Start with a simple set of features.
2. Use **Train** dataset to get insights about possible features:
   - Extract statistics or data analysis from **Train** dataset to find patterns that may be good features.
3. Create one (or a few) new features
4. Run the new feature extractor on the **Train** and **Devel** datasets.
5. Train with the **Train** dataset, and evaluate performance on **Devel**. Record the score and save the feature extractor and the vectors that produced it.
6. If the score is better, keep the new features. If it is worse, back off to best configuration found so far. Go to step 2 (or stop when the score is good enough or when no further improving is achieved)
7. Once a satisfactory configuration (features+parameters) is found, apply it to **Test** dataset, and record the score.

Machine Learning Systems Development Methodology

- NEVER look at the **Devel** or **Test** dataset.
- NEVER train with the **Devel** or **Test** dataset.
- **Train** dataset is used to learn the models.
- **Devel** dataset is used only to obtain a score and select best feature set and parameters.
- **Test** dataset is used only to obtain a final score on unseen data.

Exercise Goals

Goal 2:

Get an overall F_1 score of at least 0.6 on **Devel** dataset.

Deliverables

Prepare a report containing:

For Goal 2:

- Final version of `extract_features` function (and any other subsidiary function used by it).
- Final version of your function calling the learner (and any other subsidiary function used by it).
- Final version of your function calling the classifier (and any other subsidiary function used by it).
- Evaluator output for this version on **Devel** and **Test** datasets.

All code must be properly commented.
Self-contained Jupyter notebooks are acceptable. Notebooks don't need to be executable, use them just as a presentation layout.