Advanced Human Language Technologies

UPC

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

Outline

1. Machine Learning NERC
2. General Structure
3. Detailed Structure
   - Feature Extractor
   - Learner
   - Classifier
4. Core task
5. Evaluating Results

Session 2 - NERC using machine learning

Assignment

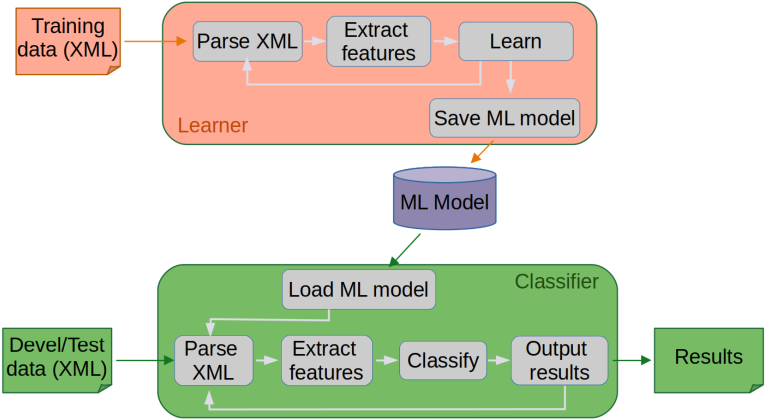
Write a python program that parses all XML files in the folder given as argument and recognizes and classifies drug names. The program must use a sequence tagging machine learning algorithm.

```
$ python3 ./ml-NER.py data/Devel/
DDI-DrugBank.d278.s0|0-9|Enoxaparin|drug
DDI-DrugBank.d278.s0|93-108|pharmacokinetics|group
DDI-DrugBank.d278.s0|113-124|eptifibatide|drug
DDI-MedLine.d88.s0|15-30|chlordiazepoxide|drug
DDI-MedLine.d88.s0|33-43|amphetamine|drug
DDI-MedLine.d88.s0|49-55|cocaine|drug
DDI-MedLine.d88.s1|82-95|benzodiazepine|drug
...
```

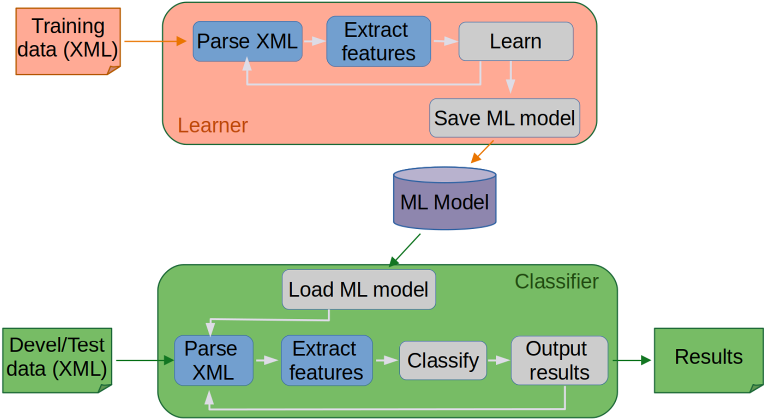
Outline

1. Machine Learning NERC
2. General Structure
3. Detailed Structure
   - Feature Extractor
   - Learner
   - Classifier
4. Core task
5. Evaluating Results

General Structure

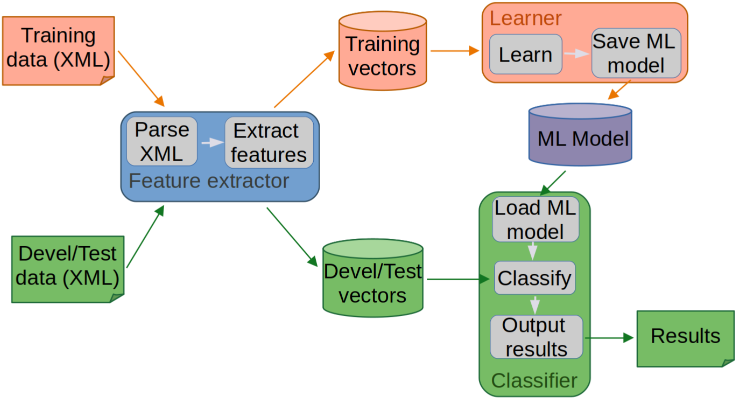


General Structure



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

1 Machine Learning NERC

2 General Structure

3 Detailed Structure
- Feature Extractor
- Learner
- Classifier

4 Core task

5 Evaluating Results

Outline

1 Machine Learning NERC

2 General Structure

3 Detailed Structure

- Feature Extractor
- Learner
- Classifier

4 Core task

5 Evaluating Results

Feature Extractor

```
def tokenize(s) :
```

Straightforwardly reuse tokenizer from rule-based NERC .

Feature Extractor

```
def extract_features(s) :
```

- Input: Receives a tokenized sentence s (list of triples `(word, offsetFrom, offsetTo)`.
- Output: Returns a list of binary feature vectors, one per token in s
- Example:

```
>>> extract_features([("Ascorbic",0,7), ("acid",9,12),
(",",13,13), ("aspirin",15,21), (",",22,22), ("and",24,26),
("the",28,30), ("common",32,37), ("cold",39,42), (".",43,43)])
[ [ "form=Ascorbic", "suf4=rbic", "next=acid", "prev=_BoS_",
"capitalized" ],
  [ "form=acid", "suf4=acid", "next=,", "prev=Ascorbic" ],
  [ "form=,", "suf4=,", "next=aspirin", "prev=acid", "punct" ],
  [ "form=aspirin", "suf4=irin", "next=,", "prev=," ],
  ...
]
```

Feature Extractor

```
def output_features(id,tokens,entities,features)
:
```

- Input: Receives a sentence id, a tokenized sentence, a list of gold entities (with their offsets) extracted from the XML, and list of binary feature vectors (one per token)
- Output: Prints to stdout the feature vectors in the following format: one line per token, one blank line after each sentence. Each token line contains tab-separated fields: sent_id, token, span_start, span_end, gold_class, feature$_1$, feature$_2$, ...

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

```
DDI-DrugBank.d553.s0 Ascorbic 0 7 B-drug form=Ascorbic suf4=rbic next=acid
prev=_BoS_ capitalized
DDI-DrugBank.d553.s0 acid 9 12 I-drug form=acid suf4=acid next=, prev=Ascorbic
DDI-DrugBank.d553.s0 , 13 13 O form=, suf4=, next=aspirin prev=acid punct
DDI-DrugBank.d553.s0 aspirin 15 21 O form=aspirin suf4=irin next=, prev=,
...
```

Outline

1 Machine Learning NERC

2 General Structure

3 Detailed Structure
- Feature Extractor
- Learner
- Classifier

4 Core task

5 Evaluating Results

Learner - Option 1: CRF

- Install and import `pycrfsuite`

  ```
  pip install python-crfsuite
  ```

- Follow this example to find out how to train a model.
- *Note: The example also extracts features, but you have this separated in another program, so you just need to load the vectors produced by the feature extractor and feed them to the learner.*
- *Note: The learner needs only the right class and the features, so you'll need to remove the other extra fields added by the feature extractor.*

Machine Learning NERC

General Structure

Detailed Structure

Learner

Core task

Evaluating Results

Learner - Option 2: 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 4 fields (*sent_id*, *token*, *span_start*, *span_end*) and the blank lines between sentences:

    ```
    cat feats.dat | cut -f5- | grep -v ’^$’ > 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 3: 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 predicting B-I-O tags for each token.

Machine Learning NERC

General Structure

Detailed Structure

Classifier

Core task

Evaluating Results

Outline

1 Machine Learning NERC

2 General Structure

3 Detailed Structure

- Feature Extractor
- Learner
- Classifier

4 Core task

5 Evaluating Results

Classifier - Option 1: CRF

- Follow this example to find out how to use a model to make predictions
- *Note: The example also extracts features, but you have this separated in another program, so you just need to load the vectors produced by the feature extractor and feed them to the classifier.*
- *Note: The classifier needs only the features, so you'll need to remove the other extra fields added by the feature extractor.*

Classifier - Option 2: Maximum Entropy

- Follow examples (and reuse code) for MaxEnt classifiers seen in class to get a B-I-O tag for each token in a sentence.

Classifier - Option 3: Your choice

- Write the necessary code to call your choice classifier and get a B-I-O tag for each token in a sentence.

Classifier - Produce output (all options)

```
def output_entities(id, tokens, classes, outf) :
```

- Input: Receives a sentence id, a tokenized sentence, a predicted class for each token, and an open output file object.
- Output: Prints on `outf` the entities in the right format: one line per entity, fields separated by '|', field order: *id*, *offset*, *name*, *type*.
- Example:

```
>>> output_entities("DDI-DrugBank.d553.s0",
        [("Ascorbic",0,7), ("acid",9,12), (",",13,13),
        ("aspirin",15,21), (",",22,22), ("and",24,26),
        ("the",28,30),("common",32,37), ("cold",39,42)],
        ["B-drug", "I-drug", "O", "B-brand", "O", "O",
        "O", "O", "O"], sys.stdout)
DDI-DrugBank.d553.s0|0-12|Ascorbic acid|drug
DDI-DrugBank.d553.s0|15-21|aspirin|brand
```

Machine Learning NERC

General Structure

Detailed Structure

Core task

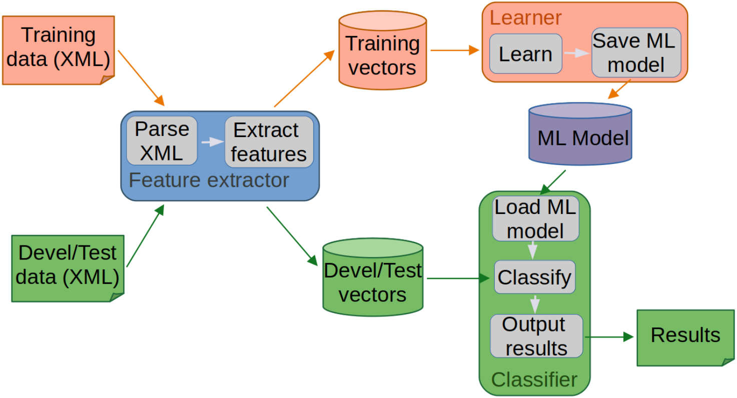
Evaluating Results

Outline

1. Machine Learning NERC
2. General Structure
3. Detailed Structure
   - Feature Extractor
   - Learner
   - Classifier
4. Core task
5. Evaluating Results

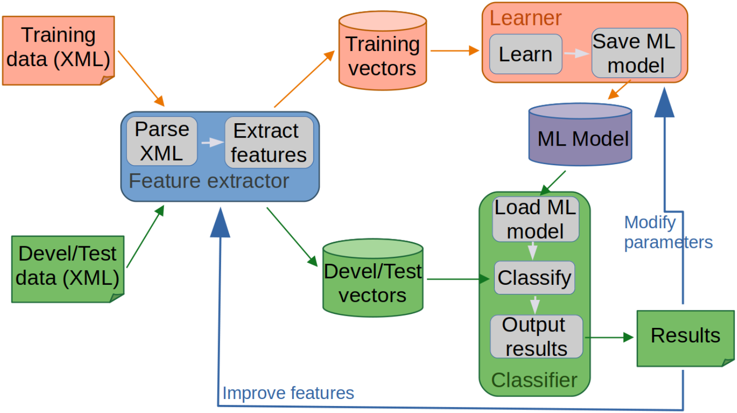
Build a good ML-based drug NERC

Strategy to follow:



Build a good ML-based drug NERC

Strategy to follow:



Outline

1 Machine Learning NERC

2 General Structure

3 Detailed Structure
- Feature Extractor
- Learner
- Classifier

4 Core task

5 Evaluating Results

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 3:
Get an overall F_1 score of at least 0.6 on **Devel** dataset using **only** information from the training dataset.

Goal 4:
Get an overall F_1 score of at least 0.7 on **Devel** dataset using also external knowledge sources.

Deliverables

Extend report from previous exercises with

For Goal 3 (ML, no external knowledge):

- Final version of `extract_features` function (and any other subsidiary function used by it).
- Final version of the function calling the classifier to get the B-I-O tags (and any other subsidiary function used by it).
- Final version of `output_entities` function (and any other subsidiary function used by it).
- Evaluator output for this version on **Devel** and **Test** datasets.

For Goal 4 (ML, using external knowledge):

- Same components, include code only for elements different than those in Goal 3
- Evaluator output for this version on **Devel** and **Test** datasets.