Master in Artificial Intelligence

Advanced Human Language Technologies

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Outline

NERC

the provided baseline recognizes and classifies drug names appearing in the sentences in given file.

```
$ python3 ./baseline-NER.py data/devel.xml index.json
result.out
$ more result.out
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
...
```

The output must be formatted like this, since it is the format expected by the evaluation script.

Data particularities

Examples

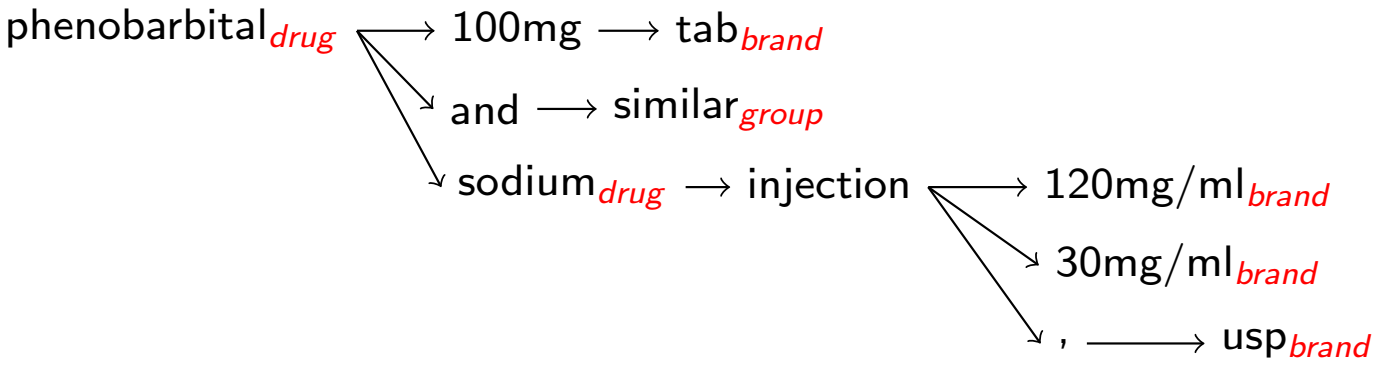
```
<document id="DDI-DrugBank.d284">
    <sentence id="DDI-DrugBank.d284.s0"
            text="If additional adrenergic drugs are to be administered by any route, they should be used with caution because the
                  pharmacologically predictable sympathetic effects of BROVANA may be potentiated.">
        <entity id="DDI-DrugBank.d284.s0.e0" charOffset="14-29" type="group" text="adrenergic drugs"/>
        <entity id="DDI-DrugBank.d284.s0.e1" charOffset="166-172"type="brand" text="BROVANA"/>
        <pair id="DDI-DrugBank.d284.s0.p0" e1="DDI-DrugBank.d284.s0.e0" e2="DDI-DrugBank.d284.s0.e1" ddi="true" type="advise"/>
    </sentence>
    (...)
    <sentence id="DDI-DrugBank.d284.s5"
              text="Although the clinical significance of these effects is not known, caution is advised in the co-administration
                    of beta-agonists with non-potassium sparing diuretics.">
        <entity id="DDI-DrugBank.d284.s5.e0" charOffset="113-125" type="group" text="beta-agonists"/>
        <entity id="DDI-DrugBank.d284.s5.e1" charOffset="132-162" type="group" text="non-potassium sparing diuretics"/>
        <pair id="DDI-DrugBank.d284.s5.p0" e1="DDI-DrugBank.d284.s5.e0" e2="DDI-DrugBank.d284.s5.e1" ddi="true" type="advise"/>
    </sentence>
    (...)
    <sentence id="DDI-DrugBank.d284.s16"
              text="In this setting, cardioselective beta-blockers could be considered, although they should be administered
                    with caution.">
        <entity id="DDI-DrugBank.d284.s16.e0" charOffset="17-45" type="group" text="cardioselective beta-blockers"/>
    </sentence>
</document>
```

Drug names vary in number of words and tokenization elements.

Outline

Simple approach

- Create an index of known groups, and search tokens in input sentence in the index
- To deal with multi-token drug names, the index must be a **prefix tree**
- Tree nodes must correspond to *tokens* according to the same tokenizer used on input data



General structure

The program expects as argument the directory with XML files to process. Results are printed to stdout

```
# Create prefix tree index of known drug names
index = DrugIndex(drugindex)
# create tokenizer
nlp = spacy.load("en_core_web_trf", enable=["tokenizer"])
# parse XML file, obtaining a DOM tree
tree = parse(datafile)
# process each sentence in the file
sentences = tree.getElementsByTagName("sentence")
for s in sentences :
    sid = s.attributes["id"].value   # get sentence id
    stext = s.attributes["text"].value   # get sentence text
    print(f"processing sentence {sid}         \r", end="")

    # tokenize text with spacy tokenizer
    tokens = nlp(stext)
    # extract entities in text
    entities = extract_entities(stext, tokens, index)

    # print sentence entities in format requested for evaluation
    for e in entities :
    print(sid,
          e["offset"],
          e["text"],
          e["type"],
          sep = "|",
          file = outf)
```

Outline

Resources

The program uses:

- An XML parser: `xml.dom.minidom` (`https://docs.python.org/3.7/library/xml.dom.minidom.html`, included in python standard libray)
- A tokenizer for English text: `SpaCy` (check `https://spacy.io/usage` if you don't have it installed)
- The evaluator module to compute performance scores (provided in the lab project zipfile).

Outline

Executing the baseline


```
$ python3 run.py
 Extracting drugs from train data
 Creating index with all known drug names
 Collecting drugs from HSDB
 Collecting drugs from DrugBank
 Collecting drugs from drugs-train
 Applying index to predict drugs
 Running baseline on devel
 Evaluating baseline on devel
 Running baseline on test
 Evaluating baseline on test
```

Results

Results on `devel` dataset

	tp	fp	fn	#pred	#exp	P	R	F1
brand	329	165	46	494	375	66.6%	87.7%	75.7%
drug	1800	648	144	2448	1944	73.5%	92.6%	82.0%
drug_n	7	16	93	23	100	30.4%	7.0%	11.4%
group	508	389	198	897	706	56.6%	72.0%	63.4%
M.avg	-	-	-	-	-	56.8%	64.8%	58.1%
m.avg	2644	1218	481	3862	3125	68.5%	84.6%	75.7%
m.avg(no class)	2719	1143	406	3862	3125	70.4%	87.0%	77.8%

Results on `test` dataset

	tp	fp	fn	#pred	#exp	P	R	F1
brand	261	180	14	441	275	59.2%	94.9%	72.9%
drug	1915	582	236	2497	2151	76.7%	89.0%	82.4%
drug_n	8	33	94	41	102	19.5%	7.8%	11.2%
group	534	456	166	990	700	53.9%	76.3%	63.3%
M.avg	-	-	-	-	-	52.3%	67.0%	57.4%
m.avg	2718	1251	510	3969	3228	68.5%	84.2%	75.5%
m.avg(no class)	2872	1097	356	3969	3228	72.4%	89.0%	79.8%

Outline

Conclusions

- A **very** simple approach based on knowledge bases (drug lists) has a pretty good performance (close to 60%macro-average, 80% micro-average).
- If we use machine learning for this task, we should aim to obtain **significantly** better results, or the additional complexity and cost won't pay off.
- Any ML project requires to start with a **baseline** that sets a threshold to calibrate cost/benefit ratio of the project. The baseline should be a simple knowledge-based or basic statistical approach.
- This will be the goal of the first lab task: **Drug NERC**.