AHLT Laboratory Session NER.1

NER baseline

In this session we will build a simple knowledge-based baseline for the first task in the DDI challenge.

We will use a skeleton that takes care of parsing XML and handling input/output formats. This skeleton will be also useful when building the ML-based approach in your project.

The program uses `xml.dom.minidom` to parse XML, and `nltk` to tokenize the text, so you'll need to have those python libraries installed

The skeleton can be found in the file `baseline-NER.py`.

The logic of the program will be described in class, though it is pretty easy to follow.

The only missing piece in the program is the function `classify_token(txt)`.

This function receives the string for a token and returns a pair (bool, string). The boolean indicates whether the word is a drug name, and the string contains its type if it is, or an empty string otherwise.

First Approach

We will start with a very basic approach to identify drugs, checking for productive token features.

For this, use the following program to dump a list of drugs and their types from the training data

```
#! /usr/bin/python3

import sys
from os import listdir
from xml.dom.minidom import parse

# directory with files to process
datadir = sys.argv[1]

# process each file in directory
for f in listdir(datadir) :
   # parse XML file
   tree = parse(datadir+"/"+f)

   # extract and print entities in file
   entities = tree.getElementsByTagName("entity")
   for e in entities :
     print(e.attributes["text"].value, e.attributes["type"].value)
```

Use this list to look for token features that indicate that the token may be a drug name, and its type. For instance:

- Tokens fully capitalized (e.g. *KERASTICK*, *DILAUDID*, *LEVSIN*) are usually drug names. Two out of three of them are of type *brand*.
- Non-capitalized words that are drugs often have particular suffixes (e.g. *-azole*, *-idine*, *-amine*, *-mycin*, etc). Words with these endings are typically drug names and most frequently of type *drug*.

Thus, you can code this simple rules into the function `classify_token(txt)` as follows:

```
def classify_token(txt):
    if txt.isupper() : return True,"brand"
    elif txt[-5:] in suffixes : return True,"drug"
    else : return False,""
```

You'll need to create a list `suffixes` that contains the most frequent suffixes for drugs found in the training data.

Improving rules

Think about similar simple patterns or features you can check on the training data and add them to your baseline.

After each new rule is added, run the baseline on the test data, and use the evaluation script to find out whether it was useful or not.

Dealing with multi-token drugs

A significant part of drug names in DDI corpus are multi-token drug names (e.g. *beta blockers*, *calcium channel antagonists*, *angiotensin converting enzyme inhibitors*, etc).

Since our baseline classifies individual tokens, multi-token drugs will suppose a loss in performance scores.

Thus, next step is dealing -in some simple way- with multi-token names.

A simple initial approach is just considering that all consecutive tokens classified as drug names by `classify_token` form a single multi-token name.

Modify the function `extract_entities` so that when several consecutive tokens are classified as drug names, only one entity is produced, spanning from the start-offset of the first token to the end-offset of the last one.

You will also need to decide which type the multi token entity will receive (e.g. the type assigned to the first token, the type assigned to the last, the most frequent type among the tokens of the entity…)

Evaluate the impact of the changes using the evaluation scripts.

Keep your baseline updated

The baseline you just built is a reference point for your later experiments. It is a simple approach requiring small effort that performs the task with a medium score.

Your lab project will consist of developing a ML approach for the same task. If your ML approach is too close to the baseline, or below it, you'll kown that either you are not doing it well, or that it is not worth using ML for this task.

You should keep your baseline updated while developing the ML system (as long as new rules added are simple and require a small effort).

For instance, if add a list of known drug names as a feature for your ML system, you should try improving the baseline with a rule using the same list, and see which impact it has.