AHLT Laboratory Session DDI.1

DDI baseline

In this session we will build a simple knowledge-based baseline for the second 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-DDI.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 `check_interaction(tokens,entities,e1,e2)`.

This function receives the tokens of the sentence, the entities in the sentence, and the ids e1,e2 of the entities we want to know whether they interact. The function returns a pair (bool, string). The boolean indicates whether the two drugs present an interaction according to the sentence, and the string contains its type if they do, or "null" otherwise.

First Approach

We will start with a very basic approach to identify interactions, checking for simple patterns.

One simple possibility is to look for specific words (such as *interacts*, *enhances*, *inhibits*, etc) between the target entities and mark the pair as an interaction if it is found.

You will need to have a look at the data to find useful words and their variants (e.g. verbs may appear in 3rd person (e.g. *inhibits*) or in infinitive (e.g. *may inhibit*). Deverbal nouns (e.g. *inhibition*) may indicate similar interactions too.

For that, use the program below, that will dump all the pairs marked as interactions, with the words in between them. You can use that list to extract words that may be relevant, and compute some frequencies about them, and about the DDI class they are likely to indicate (e.g. words like *effect*, *increase*, or *prevent* most frequently appear in *effect* DDIs, while words like *decrease*, *reduce*, or *interfere* are most common in *mechanism* DDIs).

When you have such lists, you can modifiy the function `check_interaction` to classify entity pairs depending on the words in between.

Every time you change the list of words, check the impact using the evaluation scripts.

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

# process each file in directory
for f in listdir(datadir) :

    # parse XML file, obtaining a DOM tree
    tree = parse(datadir+"/"+f)

    # 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

        tokens = tokenize(stext)

        # load and store sentence entities
        entities = {}
        ents = s.getElementsByTagName("entity")
        for e in ents :
            id = e.attributes["id"].value
            offs = e.attributes["charOffset"].value.split("-")
            entities[id] = offs

        # for each pair in the sentence, extract entities
        # and words in between
        pairs = s.getElementsByTagName("pair")
        for p in pairs:
            id_e1 = p.attributes["e1"].value
            id_e2 = p.attributes["e2"].value
            ddi =  p.attributes["ddi"].value

            if ddi=="true" and "type" in p.attributes:
                e1_start = int(entities[id_e1][0])
                e2_end = int(entities[id_e2][-1])
                print (p.attributes["type"].value+":", end="")
                for t in tokens :
                    if (t[1]>=e1_start and t[2]<=e2_end):
                        print (" "+t[0], end="")
                    print("")
```

Improving rules

When inspecting the data, it is easy to see that this first approach is oversimplistic:

- In many cases the words indicating the interactions are not between the target entities, but before or after. (e.g. *Concurrent therapy with **ORENCIA** and **TNF antagonists** is not recommended*)
- The presence of other drugs between the two target entities may indicate that the ddi-hint word we find is relating a different pair. (e.g. in the sentence *The level of **metformin** was reduced when taking **Acarbose** due to a delay in the absorption of **hydralazine***. The word *reduced* appears between the first and third drugs, but it is actually describing an interaction between the first and the second).

Add new rules to the function `check_interaction` that take into account the words not in between both target words (i.e. before the first or after the second), as well as the presence of other drugs in between.

The rules need to be simple, remember we are writing a baseline, thus we don't want to solve the problem, just assess how far can we go with a simple approach.

After adding these rules, evaluate their impact.

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).

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 to 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.

DDI:

Donar tot menys "check_interaction"

Donar regles per fer el basseline (verbs al mig, davant, darrera, patrons...)