Evaluation
Methods in CL: Lab Session

Diego Frassinelli

December 02 2019

The main goal of this lab session is to make you familiarize with the concept of *evaluation*. In this session, you will analyse the annotation generated by 32 master students from the Methods in CL class. In this handout there are **5 tasks** to perform. Please, start with task 1, 4, and 5 and if you have time perform task 2 and 3 (marked as optional).

In class we have discussed about evaluation measures. For simplicity, we have always seen cases of binary classifications (e.g., NN vs. non-NN). However, very often we have to face multi-class problems. A clear example of a task that involves many classes is POS tagging. For example, when we use the Penn-treebank annotation we have 45 tags in total[1]. Today, you will see how to deal with these situations.

Practicalities

The participation in this lab session is voluntary: you do not have to submit any result. If you want to get feedback, please ask questions during today's session.

If you find the programming part too hard for you, share a computer with a more expert student but do not be passive and try to understand what (s)he is doing and why.

To solve the tasks, you can use the programming language that you prefer. Feel free to work in small groups (max 3 people) and discuss the outcome of the different steps together.

In this handout you will find a series of review questions (**Review**) about various topics we covered in the previous lectures. Try to answer to these questions and discuss your answers with the other students you are working with.

Before you Start

All the files required for this tutorial are stored in ILIAS and on our servers. Please, download them now.

You will use the same code over and over, making small changes to it when necessary. Think about an easy way of loading in python the information contained in a table.

[1]Note: different versions of the Penn Treebank have small differences in the tagset. This count refers to the dataset we analysed in class coming from the J&M book.

For those of you that are not too experienced, a possible data structure is a list of lists. See the example below (lines starting with "#" are comments).

```
pos = line.split(",")

# each line containing the POS generated by subjects
# is split by "," and inserted into a list

subjects = list()

# we create an empty list called subjects

subjects.append(pos)

# every time we read a new line in the file
# (corresponding to a new subject) we add a new element
# to the subjects list. The new element is the corresponding pos list.
# In this way we are creating a list of lists (subjects[0][0] would give
# us the pos generated for word 1 by subject 1. Remember that we start
# counting from 0).

for x in range(0, len(subjects)):
    for y in range(0, len(pos)):
        print(subjects[x][y])

# this will go through each subject (from 0 to total) and for each
subject it will print each POS (from 0 to total).
```

Alternatively, you can use *dictionaries* or a *Pandas dataframe* [2] as data structure.

Task 1

In `IAA.csv` you can find a matrix containing the annotation of a sentence extracted from the Penn-treebank dataset (wsj_0054). The ".csv" extension suggests that this is a comma-separated-values file. If you open it with a text editor, you will see that each column in the file is separated by a comma. The first row of this file contains the sentence we are going to analyse. In the second row you can find the original annotation by expert annotators (our **gold standard**) and in the following rows, the annotations performed by 32 Methods in CL students.

1. Open the file and look at the data.
2. **Review:** what is the definition of *type* and *token*. What is the main difference?
3. Pen and Paper: Count how many types and tokens are in the sentence.

Task 2 (optional)

The first step you have to perform when you collect the annotations from different annotators is to perform a sanity check on the tags used. Did someone use tags that are not included in the official tagset? Did someone left some words without annotation?

[2] `https://pandas.pydata.org`

4. Write a script that counts the number of empty fields (marked with a "-") for each subject. (HINT: you can use a dictionary, where the keys are the subject IDs, and the values the number of empty fields).

5. Print only the subject/count pairs where the count is bigger than zero.

6. Out of the total number of subjects, how many of them missed some tags?

7. Look into the original data and see where the main problems occur.

Now we also want to perform a sanity check on the labels used. In `penntreebank.txt` you can find the list of tags provided to the annotators[3].

8. Load the content of the `penntreebank.txt` into a data structure.

9. Using the same code as before, check if the tags used by each annotator occur in the `penntreebank.txt` file. Exclude the empty cell symbol "-" from this comparison.

10. Print a list of subject/non_existing-tag (e.g., S7: VB).

As you can already see from a very short sentence, the consistency with the original tagset is essential for a good classification. Usually, when we detect such problems, we have to decide how to clean (pre-process) the data. For the purpose of this lab session, we will keep the data unmodified.

Task 3 (optional)

Now you will perform an exact match test. This test consists in checking how many subjects have a 100% match with the gold standard data. The output of this test is already a good indicator of the complexity of the task you are evaluating.

11. Write a program that compares the answers of each subject against the gold standard tags.

12. Print the subject/counts-wrong-tag pairs.

13. How many subjects obtained 100% correct answers?

14. It is interesting to check which is the tag that was missing in the cases with only one wrong answer. You can do this manually, or while you are printing the list subject/wrong.

15. Which is the most frequent mismatch among the subjects with only one mistake?

Task 4

When we are dealing with multi-category tasks, the traditional evaluation measures have to be slightly adjusted.

- **Accuracy** is: **number overlapping answers / total number of answers**

16. **Review:** what is the difference between the accuracy formula we discussed in class and this one? Why do you think we have to modify it?

- **Precision, Recall and F_1** have to be computed iteratively for each class (e.g., DT vs. non-DT, NN vs. non-NN).

[3]For simplicity, I treated the left and right double quotes as the same character.

17. Write a program that computes the accuracy by subject.

18. Compute the average accuracy.

19. Nowadays, POS taggers have an accuracy higher than 97%. How do our students perform on this task?

If you look at the modified formula for accuracy, it should look familiar. Look at the inter-annotator agreement (the version that does not correct for chance). As you can see, the two formulas look exactly the same. The only important difference lays in the assumptions behind the use of the two measurements. When we use accuracy, we are comparing the output of a model (or in our case of a subject) against gold standard data (we know what the right answer is). In the case of inter-annotator agreement, we are comparing the answers of different annotators without knowing the correct answer.

Task 5

In class we talked about two ways to compute IAA when we have more than 2 raters. One option to deal with it is to compute the IAA for each pair of subjects available (if we have 3 subjects, we will compute 3 comparisons (1-2, 1-3, 2-3)) and average the output.

20. Write a code that computes IAA (without considering chance) for each pair of subjects. Remember to exclude the row with the gold standard data from this comparison.

21. Average the answers obtained.

22. Compare this answer with the average accuracy computed before. Which one is higher? What does it tell us?

23. **Review:** what is the difference between this way of computing IAA and the use of Cohen's Kappa? Why do we usually prefer the latter?