Natural Language Processing

Politecnico di Milano - Polo di Como

Prof. Licia Sbattella - AA 2015 - 2016

"Word Sense Disambiguation"

2016-6-23 V1.1

1 Main Goal

Training and testing a Naïve-Bayes classifier for WSD.

2 The process to follow

The process is divided into three steps:

1. Choose the corpus.
2. Write the code, leveraging NLTK.
3. Train/test the model, using cross validation.
4. Write a report of the result you obtained.

3 The corpus

Choose your corpus among the ones listed in the following two web sites:

`https://wsd.nlm.nih.gov`

`https://dkpro.github.io/dkpro-wsd/corpora/`

You can choose a corpus about a non-English language, if you like it.

If you realize that the corpus you chose is too big for your computer to crunch (training time is too long), just use a subset.

4 Writing the code

The goal is to write an NLTK-based Python code that implements a WSD classifier by means of a Naïve Bayes model.

You should try to disambiguate all the word tokens that are tagged into the corpus with the correct meaning; in other words, try to disambiguate only word tokens that can be used for training the model (some corpora could provide meaning for a subset of their word tokens).

Write the code, train and test the model, as described in section 5.

5 Training/testing

Define the feature set and the related parameters.

Train and test your mode, by means of the cross-validation procedure (random or k-fold or both).

Provide results as explained in section 6, item 3.

Optional: change feature set and/or parameters, and redo the cross validation, collecting new data. Then, compare results of this new experiment with the old one, by means of the t-test. At your choice, you can compare 2 or more experiments, trying to understand the best combination of feature set and parameters.

6 How to write the Report

You can write the report in Italian or in English. The report should contain:

1. A brief description of WSD, and a brief intro to the Naïve-Bayes classifier.
2. A brief description of what you did (features you chose; how you implemented the classifier, etc.)
3. Results you obtained:
    a. For each run of the cross validation: weighted average Precision, weighted average Recall, Accuracy.
    b. At the end of the *n* runs that are part of your cross validation procedure: average Precision, average Recall, average Accuracy
    c. If you tested different models (for example, changing the feature set), you could compare the results using the t-test
4. Discussion of the results.

Remember to provide us all code and data files.

7 Info

1. You are allowed to work in a group (max two students).
2. The mark of the assignment will weight **30% of the final mark**.
3. You can present your assignment, during the oral part, and attend the written part in two different calls. We'll assign a grade to each of them, and register the final mark as soon as you pass both.
4. If you are interested in a tesina/full thesis with our group, on NLP topics, just contact us; such tesina/full thesis will be considered as a replacement for this assignment.

Problems with the tools? Typos, mistakes or unclear parts in this text? Contact *roberto.tedesco@polimi.it*

8 Deadline

Deadline: at least a week before your oral exam.
Send an e-mail, attaching the report and all related files, to:
licia.sbattella@polimi.it and *roberto.tedesco@polimi.it*
The report will be discussed as a part of the oral exam.

9 Addendum: Python

9.1 Parameters in functions and methods

Some NLTK functions and methods make use of the Python `*args` and `**kwargs` idioms to allow arbitrary number of arguments to functions and methods. Here is the meaning of such idioms:

- The `*args` will give you all parameters as a tuple:

```
def foo(*args):
    for a in args:
        print(a)

.....

>>> foo(1)
1
>>> foo(1,2,3)
1
2
3
```

- The `**kwargs` will give you all keyword arguments, except for those corresponding to a formal parameter, as a dictionary:

```
def foo(**kwargs):
    for a in kwargs:
        print(a, kwargs[a])

.....

>>> foo(name='one', age=27)
age 27
name one
```

See: http://stackoverflow.com/questions/36901/what-does-double-star-and-star-do-for-python-parameters/36908 - 36908

9.2 Integer division

In python 2.7, the `/` operator performs integer division if inputs are integers. If you want float division just use this special import at the beginning of the file:

```
from __future__ import division
```

see: http://stackoverflow.com/questions/21316968/division-in-python-2-7-and-3-3/21317109 - 21317109

Remember to insert this import, if you are using Python 2.7! For Python 3.x, it is not required as the `/` operator performs float division in any case.

9.3 Character encoding

If your corpus is encoded using Unicode, I strongly recommend to use Python 3.x. For details, see: https://docs.python.org/3/howto/unicode.html