Laboratory Project Norms for NLP courses (PLN, INLP, ATNLP)

1. Implementation should be done in Python

   Python can be obtained from http://www.python.org/ where relevant documentation and material can be found as well.
   It is a language platform free. Windows, Linux/Unix and Mac version exist.
   An excellent tutorial can be found in http://docs.python.org/tutorial/
   Whenever pieces of software in other languages (external or built by the student) are used, the corresponding interfaces from Python should be included. See Madnani (2009a) and Madnani (2009b) for interesting examples of such interfaces.

2. Use of NLTK toolbox is encouraged.

   NLTK can be obtained from http://nltk.org/ where relevant documentation and material can be found as well.
   https://sites.google.com/site/naturallanguagetoolkit/book provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more.
   Madnani (2007) presents a nice introduction to the use of NLTK with several interesting examples.

3. Other NLP packages can be used if the interfaces from Python are also included.
   a. Stanford NLP group tools: http://nlp.stanford.edu/software/
   b. Lingpipe: http://alias-i.com/lingpipe/
   c. OpenNLP: http://opennlp.apache.org/
   d. FreeLing: http://nlp.lsi.upc.edu/freeling/
   e. …

4. The organization of python files should have the following structure:
   a. All the classes used should be included in the same .py file. If the name of the application is <name>, the name of the classes file should be <name>_classes.py.
   b. All the internal functions used should be included in the same .py file. If the name of the application is <name>, the name of the functions file should be <name>_functions .py.
   c. All the external (user) functions should be included in separate .py files (scripts). In each script the classes and internal functions should be imported from the corresponding .py files.

5. For both classes and functions the habitual documentation norms should be followed.
   a. For functions, after the heading (def <name_of_the_function> (<list_of_arguments>)) a comment should be included. Comments

consist of one or more lines of text preceded and followed by a line containing only ’’’. The comment should include a short description of the function and a short description of the arguments.

b. For classes the same norms should be applied to attributes and services. An additional comment attached to the whole class has to be included. In this case comments can follow either of the two forms allowed by Python (multiline with ‘’’ or one line starting with #
c. No specific norm is needed for naming. We suggest simply using meaningful names. In the case of complex names, the first component could be lower case and the rest with the first letter uppercased (squareRoot) or alternatively separated by _ (square_root). This norm is not mandatory, it is only a suggestion.

6. For each deliverable, besides attaching python files and, if needed, data files, a documentation has to be provided. The document has to be a .pdf file containing the following (be as brief as possible):
    a. Title of the deliverable
    b. Author(s)
    c. Date
    d. Short (one paragraph) description of the content
    e. Identifiers of the attached files (classes, functions and script .py files and data files when needed)
    f. A description of the problem faced
    g. A description of the approach used for solving the problem. It is important to argument why this approach has been followed against the possible alternatives.
    h. The evaluation performed
    i. A discussion of the results
    j. A proposal of possible improvements

    Items i and j are considered extremely important for evaluation.

References

Nitin Madnani (2009a). Querying and Serving N-gram Language Models with Python. The Python Papers, 4(2). 2009.

Nitin Madnani (2009b). Source Code: Querying and Serving N-gram Language Models with Python. The Python Papers Source Codes, 1(1), 2009.

Nitin Madnani (2007): Getting started on natural language processing with Python. ACM Crossroads 13(4): 5 (2007)