Università degli Studi di Pisa

Laurea Magistrale in Informatica

Laurea Magistrale in Informatica e Networking

Advanced Programming

Middle Term Paper

Start Date: 14/11/2010
Submission deadline: 21/11/2010 (send a single PDF file to attardi@di.unipi.it)

Rules:

The assignment must be produced personally by the student, signed implicitly via his mail address.

You are allowed to discuss with others the general lines of the problems, provided that each student eventually formulates his own solution. Each student is expected to understand and to be able to explain his solution.

You are allowed to consult documentation from any source, provided that references are mentioned.

It is not considered acceptable:

- to develop code or pseudo-code with others
- to use code written by others
- to let others use someone's code
- to show or to examine the work of other students.

Violation of these rules will result in the cancellation of the exam and a report to the Presidente del Consiglio di Corso di Studio.

For the programming exercises you can choose a programming language among C++, C# and Java.

Exercise 1

Consider the problem of translating a sentence from one language to another, using a phrase table, i.e. a dictionary containing possible translations for a sequence of words. The translator (also called *decoder*) splits the original sentence into contiguous sub-phrases, looks up a possible translation and combines the translations for all sub-phrases into possible translations for the whole sentence. To avoid as much as possible string operations, the sentence to translate will be represented as a sequence of tokens and the lookup in the phrase table will be done on tokens.

Define a suitable representation for phrase tables, allowing efficient lookup: given the start position in the sentence, it must provide an `Iterator` interface, which returns one at a time, all translations of sub-phrases starting at the given position. The `Iterator` interface has a signature like this:

```
interface Iterator<T> {
    bool  HasNext();
    T Next();
}
```

Exercise 2

Implement the phrase table class and provide a reader that builds a phrase table from two parallel files: one file contains phrases in the original language, one per line, with tokens separated by spaces; the second file contains the translations, one phrase per line, of the corresponding original.

Exercise 3

Implement a decoder that will generate all possible translations for a sentence.

Exercise 4

Extend the phrase table class to contain also a probability for each translation. Revise the decoder to compute the overall probability of a sentence, defined as the product of the probabilities of the chosen sub-phrases and to return just the sentence with the highest probability.

Exercise 5

Illustrate bounded polymorphism and explain why it is necessary to introduce it in certain languages. How is the issue dealt in C++?