Méthodes Empiriques et Langages de Script
TP 7

G. A. Musillo
musillo4@etu.unige.ch

October 24, 2006

1 Exercice 1: Une classe pour les *heaps*

En Perl, une classe n'est rien d'autre qu'un *package* qui met à disposition d'un client un constructeur d'instances ou d'objets de cette classe. Typiquement, un objet est une référence sur un tableau associatif. Les clés de ce tableau associatif renvoie aux champs ou attributs de l'objet. Pour associer une référence à un module (autrement dit, pour construire un objet d'une classe), Perl utilise l'opérateur `bless` qui prend une référence et un nom de module comme arguments. Voici, par exemple, une classe pour des objects qui peuvent être enregistrés dans une queue de priorité:

```
#!/usr/bin/perl
use strict; use warnings;
package PriorityQueueObject;
sub new { # constructor
        my $class = shift;
        my $priority = shift;
        my $object = shift;
        my $pqo = {
         _priority => $priority,
         _object => $object
        };
        bless $pqo, $class;
        return $pqo;
}
sub priority {
        my $pqo = shift;
        return $pqo->{_priority};
}
sub object {
        my $pqo = shift;
```

```
        return $pqo->{_object};
}
1;
```

Et voici comment un client pourrait utiliser la classe `PriorityQueueObject`:

```
#!/usr/bin/perl
use strict; use warnings;
use PriorityQueueObject;
my $obj = 'Google';
my $pqo = PriorityQueueObject->new(128, $obj);
print $pqo->object, ":", $pqo->priority, "\n";
```

Etant donnée la classe `PriorityQueueObject`, on vous demande de programmer une classe `PriorityQueueHeap` qui implémente les queues de priorité au moyen des *heaps* (littéralement, des *tas*). Vous trouverez une description des queues de priorité et de leur implémentation en Oberon dans le document `http://www.latl.unige.ch/mels/PriorityQueueHeap.pdf`, réalisé par Luka Nerima. Il vous suffit donc de traduire le code orienté-object Oberon en code orienté-object Perl. Vous devez également écrire un programme Perl qui test votre classe `PriorityQueueHeap`.

2 Exercice 2: Commentaire des modules *Sexpr.pm* et *SexprNode.pm.*

Les documents `http://www.latl.unige.ch/mels/SexprNode.pm` et `http://www.latl.unige.ch/mels/Sexpr.pm` implémentent deux modules sans commentaires qui permettent de construire l' arbre n-aire correspondant à la notation parenthésée d'une structure syntaxique. On vous demande de commenter ces deux modules. Vos commentaires doivent montrer que vous avez compris le code de ces deux modules. Pour vous aider, téléchargez les fichiers `http://www.latl.unige.ch/mels/sexpr.dat` et `http://www.latl.unige.ch/mels/sexpressions.pl`, éxécutez `cat sexpr.dat |./sexpressions.pl` et observez la sortie.

3 Exercice 3: un classifieur Naive Bayes pour la classification de documents textuels

On vous demande de programmer un classifieur bayésien naïf capable d'apprendre à classer des documents textuels. Le pseudo-code que vous devez implémenter en Perl vous a récemment été présenté par Paola. Vous pouvez consulter

le document (`http://www.latl.unige.ch/mels/cours7-bayes.pdf`) qui décrit ce pseudo-code.
Le pseudo-code de Paola spécifie deux procédures: la procédure d'apprentissage et la procédure de classification. Bien que le pseudo-code de ces deux procédures soit correcte, son implémentation exige quelques modifications. En effet, l'algorithme bayésien naïf demande de calculer un produit dont les facteurs sont des probabilités comprises entre 0 et 1. Or, le produit de nombres compris entre 0 et 1 pourrait bien être plus petit que le plus petit nombre qu'une machine puisse représenter. Pour résoudre ce problème d'*underflow*, il suffit de substituer à une probabilité son logarithme (la fonction logarithme étant croissante, maximiser un produit de probabilités équivaut à maximiser son logarithme). Rappelez-vous que la classe retournée par le Naive Bayes est la classe c qui maximise le produit

$$P(c) \times \prod_i P(w_i|c)$$

Or le logarithme d'un produit de facteurs est la somme des logarithmes de ces facteurs. Par conséquent, votre programme devra retounrner la classe c qui maximise la somme

$$\log P(c) + \sum_i \log P(w_i|c)$$

Les données que vous utiliserez sont stockées dans l'archive

`http://www.latl.unige.ch/mels/tp7_newgroups.tar.gz`.

Cette archive contient des messages de newsgroups. Sauvegardez cette archive et désarchivez-la au moyen de la commande `tar -xzvf tp7_newgroups.tar.gz`. Il en résultera 7 répertoires: `comp.sys.mac.hardware`, `comp.windows.x`, etc. Chacun de ces répertoires correspond à une classe de newsgroups qui contient des messages. Partionnez ces messages en messages d'entraînement (99 % des messages de chaque répertoire) et messages de test (1 % des messages de chaque répertoire). Les messages n'ont pas été prétraités: ils contiennent donc des mots (ou d'autres signes) qui ne sont pas utiles à la classification. Tâchez d'éliminer ces mots ou signes inutiles.
Rapportez également toutes les mesures de performance qui vous paraissent les plus pertinentes pour évaluer votre classifieur.
Vous devez me faire parvenir votre code et vos résultats au plus tard le 17 janvier 2007 à midi.