Ecole Polytechnique de Louvain

Informatique II

Rapport de projet

Antoine Gennart

Réalisé le 26 novembre 2015.

Dans le cadre de ce projet, il nous a été demandé de réaliser un jeu "Qui oz-ce" du même style que le célèbre jeu "Qui est-ce" . Notre jeu part d'une base de données sur les différents membre de l'équipe des Diables Rouges de la coupe du monde 2014, contenant des questions et des réponses permettant de les identifier.

Il nous a été demandé de réaliser avant tout une solution de base qui se diviserait en trois étapes.
La première, **lire la base de donnée**, nous a été fournie. Les deux étapes qu'il nous restait à implémenter étaient **construire un arbre de décision pour guider la tache suivante** et **poser les questions au joueur humain pour trouver la personne choisie**.

1 Construire un arbre de décision (BuildDecisionTree)

Tout abord, nous avons implémenté la construction d'un arbre basique, qui prendrait en compte les questions dans leur ordre d'apparition. Cela nous a permis de comprendre la structure fondamentale que devait prendre cette fonction, et donc cet arbre.

La fonction prend la **database** et une **liste de questions** en argument. Dans le cas simple, elle parcourt dans l'ordre la liste de questions. Pour chacune d'entre elles, elle parcourt toute la database et construit une **ListFalse**, contenant les noms des joueurs dont la réponse à la question actuelle est **false**, ainsi qu'une **ListTrue**, contenant les noms des joueurs dont la réponse à la question est **true**. La première question posée constitue le noeud principal à partir duquel l'arbre de décision va être construit.
Cet arbre sera ensuite construit par **appels récursifs** de la fonction `{BuidDecisionTreeAcc ..}` avec à chaque appel la question suivante. A **droite** de toute question (=noeud), la fonction sera appelée avec la **liste true** à la place de la **database**, alors qu'à **gauche** elle sera appelée avec la **liste false**.

Une fois que toutes les questions ont été posées, c'est à dire que la **"question" est nil**, la fonction **renvoie une liste** contenant tous les joueurs encore présents dans la **database actuellement prise en compte** par celle-ci.

ListOfQuestions = [Question1 Question2 Question3 ...]

Question 1

Nous devons implémenter deux fonctions principales : **BuildDecisionTree** qui va créer un arbre de décision le plus optimal possible pour que l'ordinateur pose le moins de questions possibles pour tomber sur la bonne personne. Et **GameDriver**, qui va, à partir de l'arbre créer par le BuildDecisionTree, décider des questions à poser, et s'il n'y a plus de questions a poser, proposer une solution.

2 GameDriver

3 Extensions

3.1 Incertitude dans la base de donnée

3.2 Questions non binaires

3.3 Incertitude du joueur (true, false, I don't know

3.4 Gérer les erreurs du joueur

3.5 Bouton "OUPS" (revenir à la décision précédente)