Seminaire Technologique

Contents

1 Introduction

code cours : SemTec18

1.1 Qu'est-ce que l'intelligence artificielle ?

L'**intelligence artificielle** est l'étude du comportement intelligent

Les Origines de l'intelligence sont la convergence de différentes disciplines scientifiques. Son but est de rendre les systèmes plus intelligent (ou humains) pour les rendre plus utils.

1.2 Principes, méthodes et caractéristiques de l'IA

La base de l'intelligence artificielle réside dans sa connaissance (dans une base de connaissance). Le problème est lors de la représentation de ces données.

Le développement du traitement del'information en l'informatique s'est effectué en trois étapes. à l'origine, le traitement portait presque exclusivement sur de l'information numérique. Ensuite est arrivé le traitement alphanumérique (nombres et textes) (ex : BDD). Finalement, L'intelligence Artificielle représente ses connaissances (faits, énoncés, règles, méthodes) par l'intermédiaire d'un système de symboles à manipuler.

En Intelligence artificielle, on calcule peu mais on déduit beaucoup. La déduction (Si P implique Q et P alors Q) est fiable comparée à l'induction (si 3 le pigeon vole, si le merle vole et si le canard vole et tous sont des oiseaux alors tous les oiseaux volent). L'induction ne garantie donc pas la validité de sa conclusion.

L'apprentissage d'une intelligence artificielle, comme celui de l'Homme s'effectue via une construction d'expérience et une intégration de celle-ci dans la base de connaissance.

2 Domaine d'applications

Il existe de nombreux domaines dans lesquels l'intelligence artificielle peut être utilisée. On prendra notament les exemples des jeux, des démonstrations logiques, de la perception (vision, parole), de la résolution de problèmes (systèmes experts), la générations de plans.

Un système intelligent fonctionne en 3 parties. La première est la perception (reconnaissance de la parole, des écrits ou des images). la seconde est la faculté intellectuelle (connaissance, déduction, induction, plannification et apprentissage). La dernière est l'expression de la solution obtenue, que ce soit par le biais de la

génération, la synthèse ou encore le contrôle. Le système est capable d'exprimer le résultat de sa recherche.

3 Les Systèmes Experts

Un expert est une personne ou une entitée avec des connaissance et de l'expérience. L'expérience est difficille à transmettre. Cependant, les connaissances sous la forme d'une base de donnée/connaissance peuvent être copiées facilement par un système informatique. L'expérience est un procédé, une méthode de traitement de l'information. On peut parler ici d'une forme de raisonnement par le biais de règles.

Ces règles peuvent êtres appliquées à un cas particulier. Elles permettent de déduire une conclusion si des conditions sont satisfaites. Ou encore d'enchaîner des règles ($A \implies B$ et $B \implies C$ alors $A \implies C$).

Cette nouvelle forme de "raisonnement" informatique nécéssite d'abandonner les languages procéduraux classiques (boucles et affectations) qui sont devenur inapropriés pour ce genre de tâches et de se tourner vers des langages axés sur la déduction et le résonnement mathématique.

4 Connaissance Procédurale VS Connaissance Déclarative

La **Connaissance Implicite** est une connaissance procédurale, elle décrit "comment faire".

La **Connaissance explicite** est une forme **déclarative** de connaissance (règle pour un cas particulier). Sa modification est aisée et elle est indépendante de l'application, ce qui la rends réutilisable. Les connaissances déclaratives peuvent être étendues par raisonnement mathématique.

5 Prolog

Le nom de prolog vient de **Pro**gramming in **lo**gic. Le but de la programmation logique est de parvenir à trouver une solution sur base de connaissances de départ (connues). Le prolog fonctionne donc par déductions logiques successives pour trouver si oui ou non une phrase de départ est valide ou non. Toute la puissance du programme écrit résidera donc dans la complétude de la **base de connaissance** fournie au départ.

6 Base de connaissances

Une base de connaissances est constituée d'un **ensemble de clauses**. Une clause est soit un **fait**, soit une **règle**.

> Tout *fait* et toute *règle* Prolog se terminent par un point.

6.1 Faits

Le prédicat P possède n arguments (=termes).

$P(A1, ...An)$

Un **terme** est dans le domaine standard (integer, real, char(ex :'c'), symbol(ex : pierre, paul, 'Jules'). Le programeur peut aussi en définir d'autres.

Une **Variable** débute par une lettre majuscule ce qui les différencie des prédicats et constantes.

> Attention, Les majuscules sont réservées aux variables. Il est donc indispensable de mettre les termes entre " si on veut leurs mettre une majuscule.

> en prolog, les fonctions n'existent pas au sens mathématique (fonction mathématique qui retourne une valeur).

Un **fait** permet de représenter différents types de connaissances :

- La **propriété d'un objet**
  - prof(dubisy).
- La **relation entre objets**
  - frere(bob,marc).
- Une **appartenance à une classe**
  - femme(dubisy)
- Une **ligne d'une table relationnelle**
  - emprunt ('suicid squad', Dheur, '11 Janvier 2000')
- Une **probabilité**
  - suicide(antoine, examens, 0.99)
- Une **fonction classique**
  - somme(100,1,101)

En prolog, les fonctions à N arguments se traduisent par des prédicats à N+1 arguments. Ce dernier arguement étant réservé pour la réponse.

L'utilisation de variables dans les faits permets de résumer plusieurs faits (ex : aime(julie,X) julie aime tt le monde et elle meme aussi)

6.2 Règles

La forme générale d'une règle se présente sous la forme C:-a,b,c,d

$Conclusion : -Conditions$

Si toutes les conditions sont vraies, alors la conclusion sera vraie aussi. Dans le cas d'une règle ou A ou B réalisent la conclusion on va séparer les deux conditions en deux règles distinctes.

> si a(x) alors (c(x) ou d(x)) est impossible. le programeur doit donc trouver des conditions permetant de différencier C de D.

6.3 Procédures

En prolog, les procédures sont un ensemble de clauses (faits ou règles) qui définissent la même relation. Attention, si l'ordre entre les procédures n'a pas d'importance, il est important de mettre les procédures dans l'ordre. Ceci est plus performant et évite le bouclage. Les faits doivent donc apparaître en tête de procédure.

7. Le moteur d'inférence de Prolog

7.1 Les Questions

Le moteur d'inférence de Prolog exploite la base de connaissance en la questionnant.

> Attention, l'ordre compte dans les relations. questions :

- rel(a,b) : est ce que a à la relation 'rel' avec b ?
- rel(X,b) : qui à la relation 'rel' avec b ?
- rel1(X,b),rel2(X,c) : Qui à la relation 'rel1' avec b et la relation 'rel2' avec c ?
- rel(_,b) : est ce que quelqu'un à la relation 'rel' avec b ?

Le "_" peut être utilisé à la place d'une variable. il s'agit d'une **variable anonyme** et elle ne possède aucune valeur de retour.

En prolog, une question est un déclancheur d'induction ou d'inférence. Le but du système est alors d'obtenir un résultat "satisfaisant" (satisfaisant car les données peuvent être incomplètes. Ses réponses seront alors oui, ce qui signifie qu'il à été en mesure de déterminer que c'était vrai via des déductions sur les données de la base de connaissances. ou un Non, qui signifie soit qu'il n'a pas assez d'informations, soit que c'est effectivement faux.

7.2 Le Backtracking (=Rétro-parcours)

Le raisonnement d'un moteur d'inférence peut être comparré à un arbre. Il existe deux types de noeuds dans cet arbre les "et" et les "ou". On effectue une recherche exhaustive parmi tous les faits et règles jusqu'à obtenir une correspondance par un jeu de substitutions. Lorsqu'un backtracking échoue, on remonte jusqu'au neud valide et on réésaye avec un autre jeu de substitutions.

Un Objectif est dit réussi si il s'unifie à une fait ou s'unifie à une conclusion de règles dont tous les antécédents réussisent.

Règles d'unification de deux prédicats :

- Ils ont tous deux le même symbole de prédicat
- Ils ont tous deux le même nombre d'arguments
- Leurs termes s'unifient deux à deux.

Règles d'unification de deux termes :

- Une constante peut s'unifier à :
  - la même constante,
  - n'importe quelle variable.
- Une variable peut s'unifier à :
  - n'importe quelle constante,
  - n'importe quelle autre variable.

L'ordre de considération des règles (/faits) est statique, c'est-à-dire prédéterminé :

- Du haut vers le bas pour les faits et règles.
- De gauche à droite pour les conditions d'une règle.

En cas d'échec dans l'établissement d'un objectif (X), deux traitements sont possibles selon le type d'objectif qui échoue * Si X est un objectif OU (càd le fils d'un nœud OU) * On passe à l'objectif (frère) suivant, c'est-à-dire la règle suivante qui a le même conséquent (même conclusion) * Les instanciations de variables établies en vue de démontrer l'objectif qui a échoué sont annulées, et uniquement celles-là

- Si X est un objectif ET (fils d'un nœud ET)
  - On "rétro-parcourt" jusqu'à l'objectif (Y) qui avait nécessité la plus récente instanciation de variable
  - On défait cette instanciation-là, et uniquement celle-là
  - On repart de l'objectif Y qu'on réessaye avec une autre valeur d'instanciation pour la variable défaite.

8 Récursivité

Une procédure est dite réccursive si elle se rapelle elle même. Le principe d'une fonction récursive est de réduire la complexité d'un problème en le décomposant jusqu'à arriver à un cas trivial (cas de base). Toute procédure prolog doit donc comprendre deux clauses. la première clause est le cas de base, celle-ci doit être placée en tête de procédure pour permettre d'éviter un bouclage et un "Stack Overflow". On fait ensiute appel à une clause de type récursive $(p(X,Y) : -..., p(R,S), ...)$

9 Suppléments

=,<,>=,<= sont utilisées pour deux arguments (ex : $X <= 10$)

not(A=B) vérifie que A est bien différent de B

factorielle (X,Y) il serait faux de mettre une fonction sans un paramètre de retour (ici Y).

Définitions et exemples

Déduction logique : si P *implique* Q, et que P est *vrai*, alors Q est forcément *vrai*.
Induction Mathématique : Si A implique P, B implique P, ... et que tous (A,B,...) font partis de l'ensemble Z, alors on peut en conclure que Z implique P.