Compte-rendu IA01

Projet n°3

Sarah Richard et Florian Guyet ▸ 05/01/2015

Sommaire :

Compte-rendu IA01

Projet n°3

Introduction

L'objectif de ce TP est de développer un système expert d'ordre 0+ depuis sa phase d'expertise jusqu'à sa phase d'utilisation.

D'après la définition du cours de IA01, un système expert est un programme informatique qui utilise une représentation de l'expertise humaine pour accomplir efficacement des tâches habituellement réalisées par un expert humain. Le fait qu'il soit d'ordre 0+ signifie que les différents faits seront représentés par un couple (attribut, valeur) ou (objet, attribut, valeur).

Un système expert est composé de trois parties : une base de règles qui est le produit de l'expertise, une base de faits renseignée par l'utilisateur et un moteur d'inférence utilisant la base de règles pour aboutir à une expertise.

Nous allons, dans une première partie, présenter le sujet de notre système expert, puis dans une deuxième partie nous traiterons de l'élaboration des moteurs d'inférence pour montrer, dans la partie suivante, un exemple d'utilisation de notre système expert. Finalement, nous aborderons les limites de notre système et les améliorations à apporter.

Description du sujet du système expert

Choix du sujet

Nous avons tout d'abord pensé à faire un système expert indiquant dans quel pays du monde nous nous trouvions. Cependant nous nous sommes vite aperçus qu'il serait compliqué d'être exhaustif et qu'il y avait beaucoup de paramètres à prendre en compte.

Nous avons donc affiné notre sujet qui peut maintenant être décrit par l'énoncé suivant :

« Vos amis ont décidé de vous surprendre ! Ils sont arrivés chez vous à l'improviste et vous ont emmené dans leur voiture. Ils vous ont ensuite bandé les yeux et bouché les oreilles afin que vous perdiez tout repère. Rapidement, vous vous êtes endormi.

Vous venez de vous réveiller, un mot à la main :

"Surprise ! Nous t'avons déposé dans une ville avec pour seules affaires un sac de rechange, un thermomètre et un système expert sur l'ordinateur à ta gauche. Ton but va être d'observer ton environnement afin de trouver dans quelle ville tu es, à l'aide du système expert.

On te donne tout de même un indice, tu te trouves dans l'une de ces vingt villes : Compiègne, Paris, Shanghai, Tokyo, New York, Los Angeles, Tombouctou, Johannesburg, Canberra, Kuala Lumpur, Rio de Janeiro, Mexico, Moscou, Llanfairpwllgwyngyllgogerychwyrn-drobwllllantysiliogogogoch, Prague, Marrakech, Honolulu, Pyongyang, Bombay ou Londres. "

Vous allumez l'ordinateur et accédez au système expert. »

Délimitation du sujet

Grâce à notre système expert, nous pouvons trouver dans quelle ville nous nous situons. Cependant, compte tenu du nombre incommensurable de villes qui existent dans le monde, nous avons décidé de ne prendre en compte qu'une vingtaine de villes réparties dans les différentes régions du monde.

Pour savoir dans quelle ville nous nous trouvons, nous avons pris en compte cinq paramètres qui sont : la température en janvier, l'ethnie majoritairement présente, le type d'écriture, le côté de la route où l'on conduit et la langue, lorsqu'il est possible de la reconnaître (nous supposons aussi que l'utilisateur est capable de reconnaitre au moins l'espagnol, le portugais et l'anglais). Dans certains cas, nous avons aussi dû ajouter les paramètres suivants : présence de bouche de métro, couleur de la pièce unitaire du pays, présence d'accent sur les consonnes.

Ainsi notre système expert a 20 états finaux qui sont les différentes villes où il est possible de se trouver. Les faits entrés dans la base de faits sont les paramètres cités précédemment.

L'expertise

Pour pouvoir créer notre base de règles, nous avons dû nous renseigner sur divers sites afin de trouver toutes les informations nécessaires sur les villes. Nous avons regroupées ces informations dans le tableau en annexe.

Nous avons gardé en paramètres, ceux qui nous ont semblé les plus importants et qui permettent une différenciation facile et nette.

Nous avons pu trouver les températures et climats en janvier de chacune des villes grâce à ce site qui référence les températures maximales et minimales par mois : *http://www.quandpartir.com/meteo/shanghai-idville-170.html*

Pour le côté de la route où l'on conduit, nous avons utilisé ce site qui référence tous les pays du monde : *http://whatsideoftheroad.com/*

Pour la description de la monnaie nous nous sommes renseignés sur un site qui indique la monnaie du pays et ce à quoi elle ressemble : *http://www.monde-en-pieces.com/*

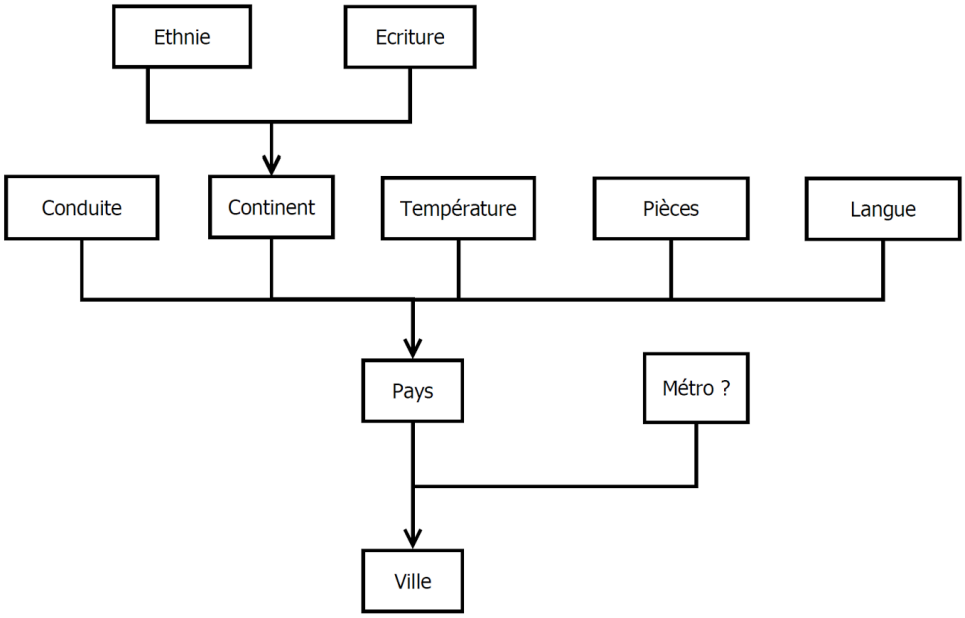
En ce qui concerne les langues, nous avons utilisé le site suivant, lorsque les pays comportaient plusieurs langues parlées, nous nous sommes renseignés sur les villes pour savoir laquelle était la plus parlée dans la ville : *http://www.infoplease.com/ipa/A0855611.html*

Nous avons utilisé ce site pour connaître l'ethnie principale des différents pays : *https://www.cia.gov/library/publications/the-world-factbook/fields/2075.html*. Puis nous avons généralisé les ethnies en cinq grands groupes : Africain, Arabe, Asiatique, Caucasien et Sud asiatique.

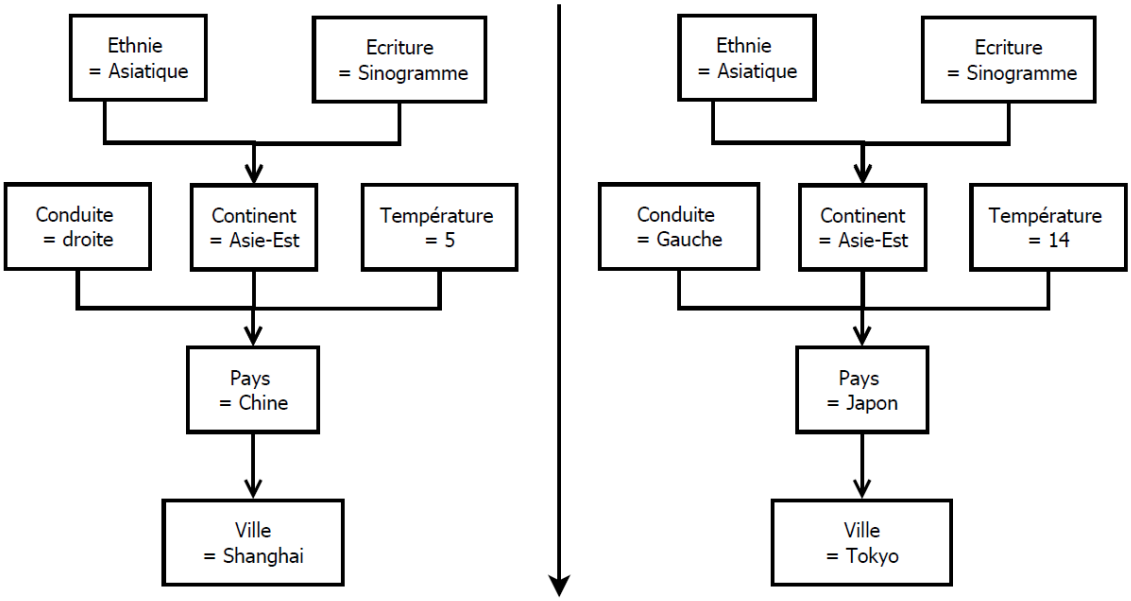
Finalement, pour différencier les différentes écritures des pays et trouver leur alphabet, nous avons utilisé ce site : *http://www.omniglot.com/writing/langalph.htm*

Mise en place de la base de règles

Pour mettre en place une base de règles, nous avons tout d'abord établi l'arbre de recherche suivant, qui nous permettait de différencier toute les villes :



Nous pouvons prendre deux exemples qui utilisent cet arbre de recherche :



Nous pouvons voir que tous les paramètres ne seront pas nécessaires pour trouver une ville, et l'arbre de recherche diffère selon les cas. Cependant, nous cherchons, lorsque c'est possible, à passer par les étapes suivantes: trouver le continent, trouver le pays, puis trouver la ville. Cela nous assure une profondeur supérieure ou égale à 3.

Le moteur d'inférence

Chaînage avant en largeur d'abord

Fonctions de service

Afin de mettre en place notre moteur, nous avons utilisé plusieurs fonctions de service. Ces fonctions de service permettent une meilleure compréhension du code, on évite ainsi de trop charger le code du moteur d'inférence.

La fonction *premisses* nous permet de récupérer les prémisses de la règle entrée en paramètre. Ces prémisses sont réorganisées pour avoir une syntaxe correspondant au lisp.

La fonction *reorganiser_p* permet de changer la structure des prémisses afin qu'elles aient une structure correspondant au lisp. Au départ, on a (langue = français) et après utilisation de la fonction, on aura (= langue français)

La fonction *conclusion* récupère les conclusions de la règle entrée en paramètre. La syntaxe de cette conclusion est réarrangée pour pouvoir correspondre à celle du lisp.

La fonction *reorganiser_c* permet de réorganiser la syntaxe de la conclusion entrée en paramètre afin qu'elle corresponde à la syntaxe de lisp. On aura ainsi (etat = Hawai) qui deviendra (= etat Hawai).

La fonction *demander* permet d'imprimer une question et récupère la réponse de l'utilisateur à cette question. Elle est utilisée pour compléter la base de fait lorsque des informations sont à préciser.

La fonction *executer* ajoute les règles exécutée à la liste des règles exécutée et ajoute les conclusions de la règle à la base de fait.

La fonction *premisses-BF* permet de vérifier si un fait est une prémisse d'une règle. Si c'est le cas, on retourne 1, sinon on retourne 0.

Moteur d'inférence

Le moteur d'inférence du chaînage avant nous permet de trouver la ville dans laquelle la personne se trouve grâce aux faits qu'elle entre en paramètres. Le moteur va tourner jusqu'à ce qu'il ne puisse plus déclencher de règles. Ainsi, si plusieurs villes correspondent aux faits entrés par l'utilisateur, elles lui seront proposées. L'utilisateur pourra ensuite utiliser ses connaissances sur les deux villes proposées pour en déduire laquelle est la bonne.

Le moteur d'inférence va parcourir la base de fait, s'il trouve un fait tel que (= ville Nom_Ville) il va renvoyer Nom_Ville qui sera la ville dans laquelle on se trouve, sinon il indiquera que la ville ne se trouve pas dans la base de données.

Notre moteur d'inférence est en largeur d'abord. Il ne teste pas les chemins possibles un par un, au contraire, il parcourt la base de règle et à chaque nouvelle règle dont les prémisses correspondent, il ajoute sa conclusion à la base de fait. Ainsi, on explore plusieurs chemins en parallèle.

Chaînage arrière en largeur d'abord

Fonctions de service

Tout comme le moteur en chaînage avant, nous avons utilisé plusieurs fonctions de service pour alléger le code du moteur en chaînage arrière. Certaines ont déjà été utilisées précédemment (*premisses* et *conclusion*). Voici les nouvelles fonctions de services que nous avons utilisés:

La fonction *conclusion-BF* permet de vérifier si un fait est une conclusion d'une règle. Si c'est le cas, on retourne 1, sinon on retourne 0.

La fonction *executer_reverse* ajoute les règles exécutées à la liste des règles exécutées et ajoute les prémisses de la règle à la base de fait. (à l'exception des prémisses qui mènent à la conclusion "pays de type latin", car plusieurs langues mènent à cette conclusion)

La fonction *question-BF* est utilisée lorsque l'on tombe sur une règle dont la conclusion est une question. La fonction permet de trouver la réponse correspondant à la question posée si elle est présente dans la base de fait et retourne 1. Elle retourne 0 si la réponse à la question n'est pas dans la base de fait.

Moteur

Le moteur d'inférence du chaînage arrière permet à l'utilisateur d'essayer de deviner dans quelle ville il se trouve. Le moteur lui donnera les valeurs des paramètres de cette ville. L'utilisateur pourra donc vérifier si sa supposition était bonne.

Le moteur va tourner et déclencher toutes les règles dont la conclusion est dans la base de fait, jusqu'à ce qu'il ne puisse plus trouver de règle à déclencher. Le moteur affichera ensuite tous les paramètres que l'utilisateur pourra vérifier.

Notre moteur d'inférence est en largeur d'abord. Il ne teste pas les chemins possibles un par un, au contraire, il parcourt la base de règle et à chaque nouvelle règle dont les prémisses correspondent, il ajoute sa conclusion à la base de fait. Ainsi, on explore plusieurs chemins en parallèle.

Utilisation du système expert

Interface avec l'utilisateur

Afin de simplifier l'utilisation de notre système expert, nous avons créé une interface entre le programme et l'utilisateur. Cette interface permet de placer l'utilisateur dans le contexte et de récupérer les paramètres importants qui nous permettrons de mener à bien l'expertise. Ces paramètres sont la température, l'ethnie, l'alphabet, le côté de la route où les voitures conduisent et la langue, si elle est connue de l'utilisateur.

Ou-suis-je ?

Dans cette partie, on utilise le moteur d'inférence en chainage avant, on cherche à l'aide de ce moteur dans quelle ville nous nous trouvons. Pour commencer, il faut utiliser la fonction (ou-suis-je) qui va décrire le contexte et recueillir les informations :

```
CG-USER(22): (ou-suis-je)

"Bonjour. Bien dormi ?
Vous vous demandez ce que vous faites-là ?
Vos 'amis' vous proposent un petit jeu :
Ils vous ont emmené quelque part dans le monde.
Pour vous aider, voici la liste des villes possible :

-Compiègne
-Paris
-Shanghai
-Tokyo
-New York
-Los Angeles
-Tombouctou
-Johannesburg
-Canberra
-Kuala Lumpur
-Rio de Janeiro
-Mexico
-Moscou
-Llanfairpwllgwyngyllgogerychwyrndrobullllantysiliogogogoch
-Prague
-Marrakech
-Honolulu
-Pyongyang
-Bombay
-Londres

Afin de retrouver où vous êtes, on vous demande de trouver et d'indiquer :
la température, la langue (si possible), l'ethnie de la population locale,
le type d'écriture utilisé dans cette ville, le côté de la route ou les voitures conduisent."
"Commençons avec la température :"
```

On entre donc les paramètres demandés, et le moteur va en déduire dans quelle ville nous nous situons tout en affichant les différentes règles déclenchées :

```
"Commençons avec la température :" 5

"Quelle est l'ethnie de la population locale ?
  (asiatique, sudasiatique, caucasien, africain, arabe) " caucasien

"Quel est le type d'écriture ?  (cyrillique, abjad, sinogramme, latine) :" latine

"De quel côté de la route les voitures roulent-elles ? (droite/gauche)" droite

"Reconnaissez-vous la langue ? (oui/non) :" oui

"Entrez la langue reconnue" français

(LA RÈGLE R10 A ÉTÉ EXÉCUTÉE)
(= TYPE_PAYS LATIN)
(LA RÈGLE R23 A ÉTÉ EXÉCUTÉE)
(= PAYS FRANCE)
"Marcher pendant 20 minutes, avez-vous croisé une bouche de métro ?" non

(LA RÈGLE R24 A ÉTÉ EXÉCUTÉE)
(= RÉPONSE_FR (DEMANDER "Marcher pendant 20 minutes, avez-vous croisé une bouche de métro ?"))
(LA RÈGLE R26 A ÉTÉ EXÉCUTÉE)
(= VILLE COMPIÈGNE)

(VOUS ÊTES À COMPIÈGNE !)
```

Ainsi, l'interface avec l'utilisateur permet de récupérer les paramètres essentiels à l'expertise, ces paramètres sont ajoutés à la base de fait. Le moteur affiche ensuite au fur et à mesure les différentes règles exécutées et le fait qui est ajouté à la base de fait. Il affiche finalement le résultat de l'expertise.

Suis-je-à ?

Dans cette seconde partie, on utilise le moteur d'inférence en chainage arrière. L'utilisateur suppose qu'il est dans une ville, et il veut vérifier si sa supposition est bonne. Il aura juste à écrire (suis-je-à <VilleAVerifier>). Par exemple :

```
CG-USER(29): (suis-je-à Compiègne)

(LA RÈGLE R26 A ÉTÉ EXÉCUTÉE)
((= RÉPONSE_FR NON))
(LA RÈGLE R24 A ÉTÉ EXÉCUTÉE)
((= PAYS FRANCE))
(LA RÈGLE R23 A ÉTÉ EXÉCUTÉE)
((> TEMPERATURE -4) (< TEMPÉRATURE 7) (= TYPE_PAYS LATIN))
"-----------"
"Voici les paramètres que vous pouvez vérifier pour savoir si
vous êtes bien dans la ville demandée :"
(TYPE_PAYS = PAYS_LATIN)
(TEMPÉRATURE < 7)
(TEMPERATURE > -4)
(PAYS = FRANCE)
"Marcher pendant 20 minutes, avez-vous croisé une bouche de métro ?"
(RÉPONSE_FR = NON)
(VILLE = COMPIÈGNE)
FIN
```

Le moteur de chaînage arrière exécute toutes les règles qu'il peut pour obtenir le plus de paramètres possible, puis retourne ces paramètres pour que l'utilisateur puisse vérifier s'il est bien dans la ville rentrée en paramètre de la fonction.

Limite de notre système et améliorations à apporter

Nous avons vu dans la première partie que nous avions rapidement dû délimiter notre sujet dans le cadre de l'UV (car nous étions limités par temps). Cependant, si nous voulions améliorer notre système, nous ajouterions de nouvelles villes.

Pour que cela soit possible, il nous serait obligatoire d'étendre les paramètres considérés. Il faudrait trouver de nouveaux paramètres qui permettraient de départager deux villes qui se ressemblent. Par exemple, avec notre système, si nous rentrons les données que nous pouvons observer à Amiens, nous tomberions sur la ville de Compiègne. Il faudrait donc trouver ce qui différencie Amiens de Compiègne.

Nous pouvons penser à de nombreux paramètres tels que la hauteur des immeubles, la population, l'existence d'un cours d'eau dans la ville, etc…

Cependant ce système demanderait un travail conséquent. Il faudrait une base de données gigantesque pour stocker tous les paramètres que nous pourrions prendre en compte, pour chaque ville existante. De plus, certains paramètres utiles ne pourraient pas être visibles par l'utilisateur (ex : nombre d'habitants) et cela ne serait pas compatible avec notre sujet.

De plus, notre base de règles contient déjà 60 règles, pour une vingtaine de villes. Recenser tous les paramètres des nouvelles villes et créer les règles correspondantes demanderaient un travail colossal.

Conclusion

Ce projet nous a appris de nombreuses choses. Au tout début de ce projet, nous avons mis du temps à bien délimiter notre sujet pour que le traitement de celui-ci soit faisable dans la limite de temps demandée, et que l'utilisation de ce système expert soit justifiée. Il est difficile de trouver un sujet passionnant et se trouver des limites.

Ensuite, le recensement des paramètres que nous avons utilisés dans notre base de règles a aussi été long, mais ce travail est essentiel à l'élaboration d'un système expert. L'expertise est ce sur quoi notre base de règles est fondée, et il faut qu'elle soit suffisamment complète pour que le système fonctionne correctement.

Les moteurs d'inférence, eux, ont été relativement rapide à élaborer, grâce aux compétences que nous avons acquises au cours de l'UV IA01 ce semestre. Pour chaque règle, le moteur fonctionne de la même façon jusqu'à ce qu'il n'ait plus aucune règle à exécuter.

Ce TP a été un excellent moyen de mettre nos connaissances sur le sujet de l'Intelligence Artificielle en application. Il nous a surtout permis de comprendre la difficulté de la mise en place de systèmes experts et la dose de travail nécessaire pour effectuer un système expert.

Annexes

Villes	Pays	Continent	Monnaie	Langue
Compiègne	France	Europe	Euro	Français
Paris	France	Europe	Euro	Français
Shanghai	Chine	Asie	Yuan	Mandarin
Tokyo	Japon	Asie	Yen	Japonais
New York	Etats-Unis	Amérique du Nord	Dollar	Anglais
Los Angeles	Etats-Unis	Amérique du Nord	Dollar	Anglais
Tombouctou	Mali	Afrique	Franc CFA	Français, Songhay
Johannesburg	Afrique du Sud	Afrique	Rand	Nguni, Sotho, Anglais
Canberra	Australie	Océanie	Dollar Australien	Anglais
Kuala Lumpur	Malaisie	Asie	Ringgit	Malais
Rio de Janeiro	Brésil	Amérique du Sud	Réal brésilien	Portugais
Mexico	Mexique	Amérique du Nord	Peso mexicain	Espagnol
Moscou	Russie	Eurasie	Rouble Russe	Russe
Llanfairpwllgwyngyllgoç	UK (Pays de Galles)	Europe	Livre sterling	Anglais, Gallois
Prague	République Tchèque	Europe	Couronne tchèque	Tchèque
Marrakech	Maroc	Afrique	Dirham Marocain	Arabe
Honolulu	Etats-Unis - Hawai	Océan pacifique	Dollar	Anglais, Hawaiien
Pyongyang	Corée du Nord	Asie	Won	Coréen
Bombay	Inde	Asie	Roupie Indienne	Hindi, Anglais
Londres	UK (Angleterre)	Europe	Livre sterling	Anglais

Villes	Climat	Température janvier	éthnicité	Ecriture	Coté de la route
Compiègne	Océanique	1 à 6	Caucasien	latin	Droite
Paris	Océanique	0 à 5	Caucasien	latin	Droite
Shanghai	Subtropical humide	(2-9)2 à 9	Asiatique	sinogramme	Droite
Tokyo	Subtropical humide	2 à 10	Asiatique	sinogramme	Gauche
New York	Continental humide	-3 à 4	Caucasien	latin	Droite
Los Angeles	Méditerranéen	10 à 20	Caucasien	latin	Droite
Tombouctou	Désertique chaud	13 à 30	Africain	abjad/latin	Droite
Johannesburg	Tropical tempéré	15 à 26	Africain	abjad/latin	Gauche
Canberra	Océanique	13 à 28	Caucasien	latin	Gauche
Kuala Lumpur	Equatorial	22 à 33	Asiatique	abjad	Droite
Rio de Janeiro	Tropical	23 à 30	Caucasien	latin	Droite
Mexico	Tempéré d'altitude	9 à 21	Caucasien	latin	Droite
Moscou	Continental	-15 à -8	Caucasien	cyrillique	Droite
Llanfairpwllgwyngyllgoç	Océanique	-7 à 2	Caucasien	latin	Gauche
Prague	Continental tempéré	-4 à 1	Caucasien	latin	Droite
Marrakech	Méditerranéen	6 à 18	Arabe	abjad	Droite
Honolulu	Tropical typique	22 à 25	Asiatique	latin	Droite
Pyongyang	Continental humide	-11 à -1	Asiatique	sinogramme	Droite
Bombay	Climat tropical	19 à 30	Sudasiatique	latin/abjad	Gauche
Londres	Océanique	2 à 8	Caucasien	latin	Gauche