

Conception de jeu intelligent

LE PENDU

Equipe 09

Arthur Klein III 244 595
Maxime Leroy III 244 596
Henri Longle III 244 597
IFT2003 | 05/11/2018

Table des matières

Introduction	2
Concept et choix du jeu	3
Modélisation du problème	4
Texte explicatif du predicat	8
Les résultats.....	8
Conclusion.....	11
Grille d'autoévaluation.....	12

Introduction

Dans le cadre de ce TP, il fallait réaliser un jeu en prolog ainsi qu'une intelligence artificielle capable de jouer à ce même jeu. Nous avons choisi d'implémenter le jeu du « Pendu », le choix du jeu fut assez rapide, car les règles sont simples ce qui a facilité l'implémentation en prolog. La deuxième étape était d'avoir une intelligence artificielle capable de jouer, et par la suite lui implémenter un algorithme capable de gagner des parties simples. Cependant l'algorithme était bien plus complexe que nous le pensions, dans un premier temps il a fallu « enregistrer » un dictionnaire pour que l'intelligence artificielle puisse se repérer en fonction de son état et déterminer quels sont les lettres qui apparaissent le plus dans les mots probables. Si le mot choisi est dans le dictionnaire, l'application a de grandes chances de trouver le mot. Cependant nous n'avons pas pu implémenter la totalité de l'algorithme, dont la pondération exacte des lettres et des mots probables. De plus le dictionnaire est limité car il était impossible d'implémenter la totalité du dictionnaire français.

Concept et choix du jeu

Dans le cadre de ce TP nous avons décidé de mettre en place une intelligence artificielle capable de résoudre le jeu du pendu.

C'est un jeu très simple qui consiste à trouver un mot en devinant les lettres qui le composent. Pour se faire, le joueur qui doit deviner le mot a un nombre de chances limitées, il doit à chaque tour donner une lettre de l'alphabet, si celle-ci n'est pas présente dans le mot le joueur perd une vie (dans notre variante du jeu, le joueur a 7 vies). La partie se termine lorsque le joueur qui doit trouver le mot n'a plus de vie (défaite) ou trouve le mot (victoire).




Voici le déroulement classique d'une partie :

1. Le joueur 1 va décider d'un mot de la langue française, il va ensuite tracer un nombre de tiret correspondant au nombre de lettres dans le mot ainsi que la « potence » (représentant le nombre de vies).
2. Le joueur 2 annonce une lettre
 - a. Si cette lettre fait partie du mot, le joueur 1 va la placer autant de fois qu'elle se trouve dans le mot
 - b. Si cette lettre ne fait pas partie du mot, le joueur 1 va dessiner un trait du pendu, ce qui signifie que le joueur 2 perd une vie

Le jeu se poursuit jusqu'à ce que l'un des joueurs gagne

- Le joueur 1 gagne si le joueur n'a pas trouvé le mot
- Le joueur 2 gagne s'il a trouvé le mot

Vous trouverez ci-dessous un exemple :

Le joueur 1 choisi le mot « exemple »	
Le joueur 2 annonce « E »	_ _ _ _ _ E _ E _ _ E
Le joueur 2 annonce « A »	
Le joueur 2 annonce « T »	
Le joueur 2 annonce « L »	E _ E _ _ L E
Le joueur 2 annonce « X »	E X E _ _ L E
Le joueur 2 dit avoir trouvé le mot « EXEMPLE »	E X E M P L E

Le joueur 2 gagne

NB : Dans notre implémentation en prolog, la potence est représentée par un émoticône en ASCII, de plus les vies restantes s'affichent.

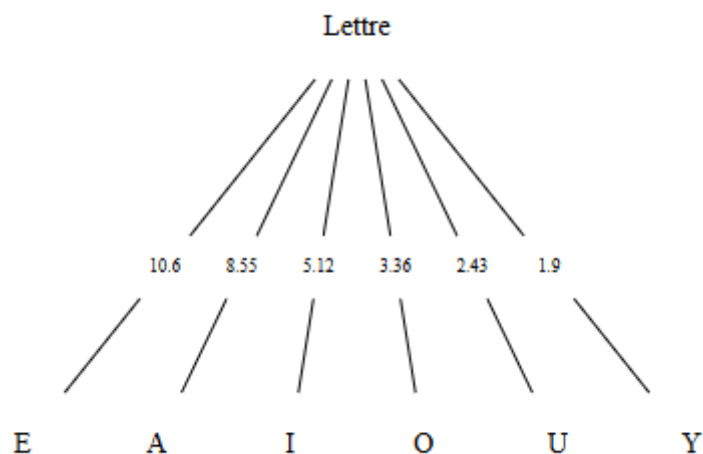
Modélisation du problème

L'état initial du problème correspond au nombre de tiret sans aucune lettre.

— — — — —

« ETAT INITIAL »

Depuis cet état initial l'intelligence artificielle ne peut presque rien faire, sa première action sera de déterminer un « état de départ » pour l'algorithme. Pour se faire elle va tenter, jusqu'à ce qu'une occurrence soit trouvée, de donner les voyelles les plus utilisées de l'alphabet et ce grâce à l'arbre ci-dessous.

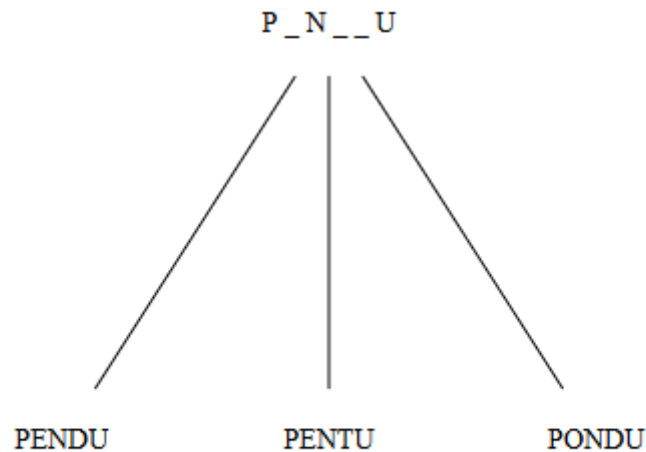


Si aucune occurrence n'est trouvée après avoir testé toutes les voyelles, le jeu est déterminé comme fini (car aucun mot de la langue française ne contient aucune voyelle).

Le poids des différentes branches a été déterminé selon l'occurrence de ces voyelles dans le dictionnaire français.

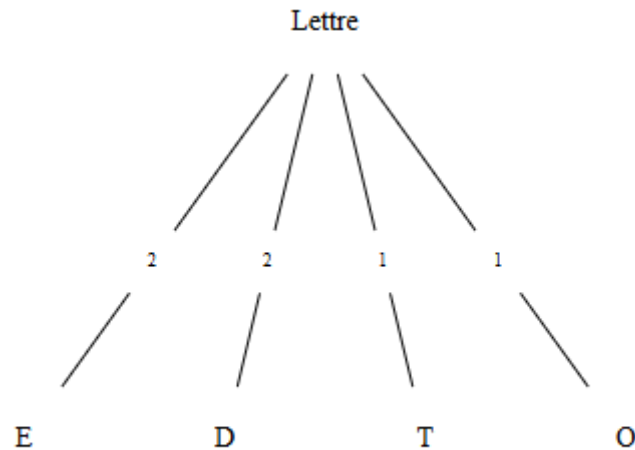
Une fois la première lettre posée, l'intelligence artificielle va évaluer le nombre de lettres bien placées afin de déterminer un nombre de mots possibles en fonction de ces mêmes lettres (heuristique).

(Dans un souci de clarté, nous imaginons que le mot est « PENDU » et que l'état actuel est « P_N__U »)



« MOT PROBABLE »

Depuis cet arbre là il est possible d'étudier le nombre d'occurrence des lettres afin de déterminer l'arbre suivant :



« OCCURRENCE DES LETTRES »

L'intelligence artificielle va continuer ainsi jusqu'à ce qu'elle trouve le mot.

Nous avons donc trois heuristiques :

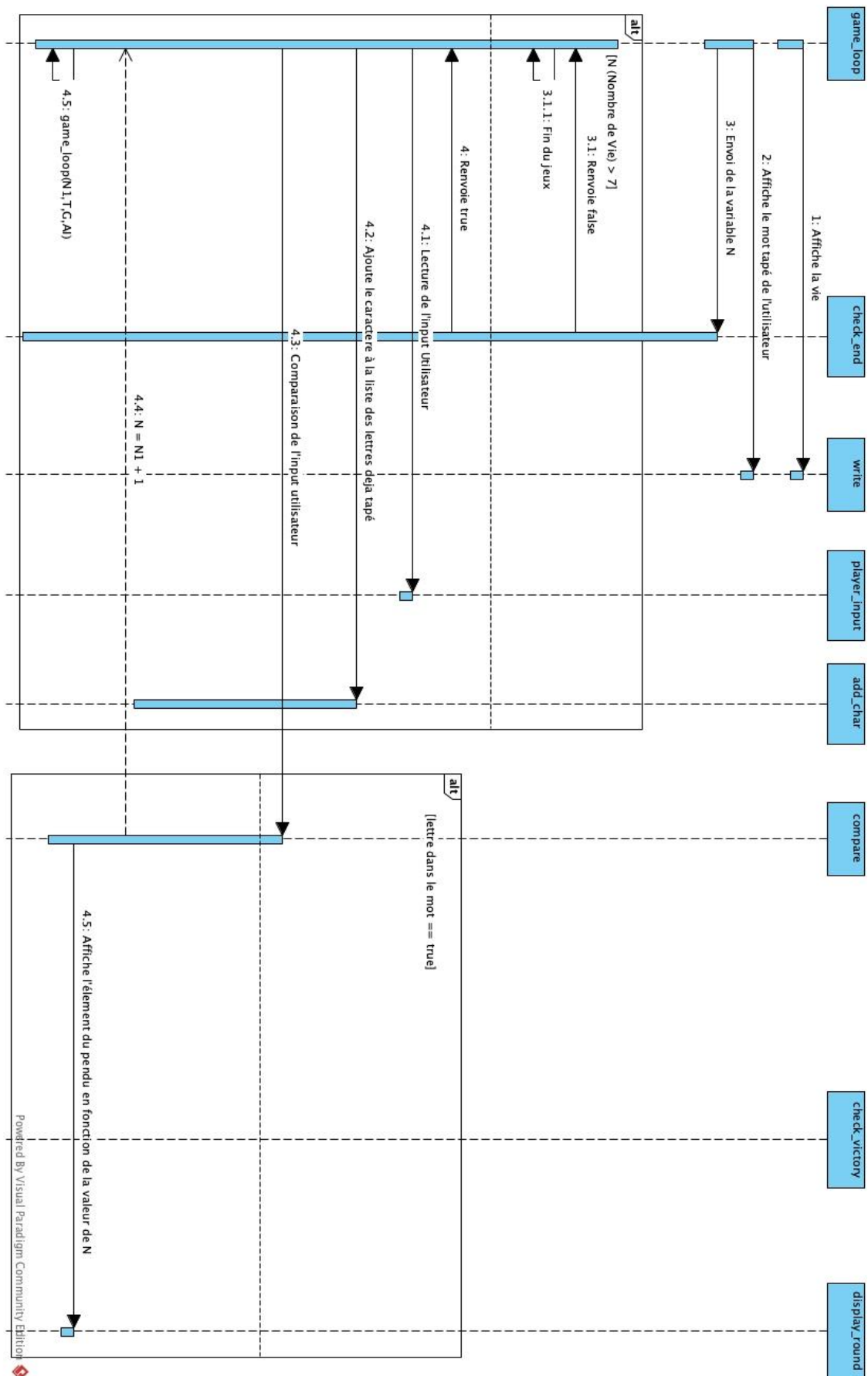
- Le poids des lettres les plus utilisées dans la langue française
- Le nombre de lettres bien placées
- L'occurrence des lettres dans les mots probables

Utilisées pour arriver à l'état final :

P E N D U

« ETAT FINAL »

EXPLICATION DU PREDICAT GAME_LOOP



Texte explicatif du prédicat

Nous avons décidé d'illustrer le prédicat de `game_loop` grâce à un diagramme de séquence. Chaque case bleue montre les prédicats appelés dans le prédicat `game_loop`. La variable `N` représente le nombre de vie restantes de l'utilisateur.

Les résultats

L'algorithme mis en place semble être le plus adapté pour la résolution de ce type de jeu. En effet, cela ressemble à la logique humaine tout en ayant les avantages d'une intelligence artificielle (la rapidité et la connaissance).

Pour ce qui est des performances, si le mot proposé est dans le dictionnaire, l'intelligence artificielle a de grandes chances de le trouver. A contrario si le mot proposé n'est pas présent dans le dictionnaire les performances de l'application sont bien amoindries ce qui réduit les chances de trouver le mot.

```
?- start_ai.  
please enter a word to find:  
|   sauver.  
life: 7  
typed:  
please type a character  
Letter choosen by AI: e  
life: 7  
typed: e  
please type a character  
Letter choosen by AI: a  
life: 7  
typed: ae  
please type a character  
Letter choosen by AI: s  
life: 7  
typed: sae  
please type a character  
Letter choosen by AI: r  
life: 7  
typed: rsae  
please type a character  
Letter choosen by AI: u  
life: 7  
typed: ursae  
please type a character  
Letter choosen by AI: v  
sauver was the word, you won !  
% Execution Aborted
```

JEU D'ESSAI AVEC UN MOT PRESENT DANS LE DICTIONNAIRE

```
?- start_ai.  
please enter a word to find:  
| anticonstitutionnellement.  
life: 7  
typed:  
please type a character  
Letter choosen by AI: e  
life: 7  
typed: e  
please type a character  
Letter choosen by AI: a  
life: 7  
typed: ae  
please type a character  
Letter choosen by AI: i  
life: 7  
typed: iae  
please type a character  
Letter choosen by AI: s  
life: 7  
typed: siae  
please type a character  
Letter choosen by AI: n  
life: 7  
typed: nsiae  
please type a character  
Letter choosen by AI: r  
/  
life: 6  
typed: rnsiae  
please type a character  
Letter choosen by AI: t  
life: 6  
typed: trnsiae  
please type a character  
Letter choosen by AI: o  
life: 6  
typed: otrnsiae  
please type a character  
Letter choosen by AI: l  
life: 6  
typed: lotrnsiae  
please type a character  
Letter choosen by AI: u  
life: 6  
typed: ulotrnsiae  
please type a character  
Letter choosen by AI: d  
/-  
life: 5  
typed: dulotrnsiae  
please type a character  
Letter choosen by AI: c  
life: 5  
typed: cdulotrnsiae  
please type a character  
Letter choosen by AI: m  
anticonstitutionnellement was the word, you won !  
% Execution Aborted
```

JEU D'ESSAI AVEC UN MOT NON PRESENT DANS LE DICTIONNAIRE

```
?- start_ai.  
please enter a word to find:  
|   xylophone.  
life: 7  
typed:  
please type a character  
Letter choosen by AI: e  
life: 7  
typed: e  
please type a character  
Letter choosen by AI: a  
/  
life: 6  
typed: ae  
please type a character  
Letter choosen by AI: i  
/-  
life: 5  
typed: iae  
please type a character  
Letter choosen by AI: s  
/_^  
life: 4  
typed: siae  
please type a character  
Letter choosen by AI: n  
life: 4  
typed: nsiae  
please type a character  
Letter choosen by AI: r  
/_^  
life: 3  
typed: rnsiae  
please type a character  
Letter choosen by AI: t  
/_^^  
life: 2  
typed: trnsiae  
please type a character  
Letter choosen by AI: o  
life: 2  
typed: otrnsiae  
please type a character  
Letter choosen by AI: l  
life: 2  
typed: lotrnsiae  
please type a character  
Letter choosen by AI: u  
/_^^_  
life: 1  
typed: ulotrnsiae  
please type a character  
Letter choosen by AI: d  
you are dead ! /_^^_-/  
% Execution Aborted  
?-
```

JEU D'ESSAI AVEC UN MOT NON PRESENT DANS LE DICTIONNAIRE

Il était aussi possible d'implémenter un algorithme qui étudie la récurrence des lettres sans passer par un dictionnaire. C'est-à-dire quelles sont les lettres les plus souvent utilisées dans la langue française en fonction de l'état actuel du jeu. Par exemple, si le mot est « A P P E L L E » et que l'état actuel est « _ P P E _ _ E », l'application devrait déterminer quelles sont les lettres les plus utilisées avant « PP ». Bien que cet algorithme se passe de dictionnaire, l'application aurait un taux de succès beaucoup plus faible. Il faudrait plus d'itération au programme pour trouver le mot, car l'intelligence artificielle serait beaucoup plus indécise. Il serait possible d'implémenter cet algorithme en complément de celui déjà implémenter pour améliorer les chances de notre programme lorsqu'un mot n'est pas présent dans le dictionnaire.

En termes d'améliorations possibles nous pourrions aussi implémenter plus de mots dans notre dictionnaire et supporter tous les alphabets existants.

Conclusion

En conclusion nous sommes satisfaits du travail effectué, nous avons pu implémenter le jeu ainsi qu'une intelligence artificielle fonctionnelle. Nous aurions aimé implémenter un algorithme plus complexe pour résoudre les mots qui ne sont pas présents dans le dictionnaire. Cependant nous restons satisfaits des résultats de notre intelligence artificielle, qui, lorsque le mot est présent dans le dictionnaire, réussit souvent à trouver le mot.

De plus l'implémentation de nouveaux mots dans le dictionnaire est très simple, il suffit d'agrandir la base de connaissance du programme.

Grille d'autoévaluation

Description du jeu choisi (10%)				
Présentation du jeu	Le but du jeu est présenté ainsi que des exemples.	Le but du jeu est présenté ou des exemples sont donnés.	Les explications et/ou les illustrations données sont incorrectes ou incomplètes.	Le but du jeu n'est pas présenté et aucun exemple n'est donné.
Description des règles	Les règles sont expliquées et illustrées.	Les règles sont expliquées ou illustrées.	Les explications et/ou les illustrations données sont incorrectes ou incomplètes.	Les règles ne sont ni expliquées, ni illustrées.
Modélisation du problème et de la solution (30%)				
Description du problème	Tous les éléments du problème sont expliqués et illustrés en utilisant l'approche par espace d'états.	Les éléments du problème sont expliqués ou illustrés en utilisant l'approche par espace d'états.	Les explications et/ou les illustrations données sont incorrectes ou incomplètes.	Les éléments du problème ne sont ni expliqués, ni illustrés.
Description de la solution	Tous les éléments de la solution sont expliqués et illustrés.	Les éléments de la solution sont expliqués ou illustrés.	Les explications et/ou les illustrations données sont incorrectes ou incomplètes.	Les éléments du problème ne sont ni expliqués, ni illustrés.
Implantation (30%)				
Fonctionnement	Le programme compile sans erreur et joue correctement.	Le programme compile sans erreur et joue correctement.	Le programme compile sans erreur mais joue incorrectement.	Le programme ne compile pas sans erreur.
Documentation	Les prédicats importants sont expliqués et illustrés.	Les prédicats importants sont expliqués ou illustrés.	L'explication et/ou l'illustration donnée est incorrecte.	Les prédicats importants sont ni expliqués, ni illustrés.
Résultats et discussion (20%)				
Discussion des jeux d'essais	Plusieurs jeux d'essais sont présentés et les résultats sont discutés.	Plusieurs jeux d'essais sont présentés mais les résultats ne sont pas discutés.	Un seul jeu d'essai est présenté et discuté.	Pas de jeu d'essai ou un seul jeu d'essai est présenté mais non discuté.
Avantages et limites	Plusieurs avantages et limites sont proposés.	Un avantage et une limite sont proposés.	Aucun avantage ou aucune limite n'est proposé.	Aucun avantage, ni aucune limite n'est proposé.
Améliorations possibles	Une amélioration possible est proposée et expliquée.	Une amélioration possible est proposée mais non expliquée.	Aucune amélioration n'est proposée.	
Appréciation globale (10%)				
Expression écrite	Le rapport ne contient aucune faute (vocabulaire, grammaire, syntaxe, etc.).	Le rapport ne contient pas plus d'une dizaine de fautes (vocabulaire, grammaire, syntaxe, etc.).	Le rapport ne contient pas plus de 5 fautes par page (vocabulaire, grammaire, syntaxe, etc.).	Le rapport contient plus de 5 fautes par page (vocabulaire, grammaire, syntaxe, etc.).
Présentation du rapport	Le format proposé est respecté. Le rapport est paginé.	Sans pagination ou Manque un élément du format	Le rapport n'est pas paginé. Il manque 2 éléments du format.	Il y a plus de 2 éléments du format qui ont été oubliés.