Projet interdisciplinaire ou de recherche

Représentation de l'environnement par un agent cognitif

Maxime Louis
Nicolas Warnet

Année 2015–2016

Projet réalisé pour les équipes Cortex et Maia du Loria

Encadrants : Boniface Yann
Dutech Alain
Jacquey Evelyne

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Louis, Maxime

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31205425

Année universitaire : 2015–2016

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Interfacer un modèle d'apprentissage du langage naturel avec des vidéos issues de la webPLM

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Nancy, le 24 mai 2016

Signature :

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Warnet, Nicolas

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31415793

Année universitaire : 2015–2016

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Interfacer un modèle d'apprentissage du langage naturel avec des vidéos issues de la webPLM

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Nancy, le 24 mai 2016

Signature :

Projet interdisciplinaire ou de recherche

Représentation de l'environnement par un agent cognitif

Maxime Louis
Nicolas Warnet

Année 2015–2016

Projet réalisé pour les équipes Cortex et Maia du Loria

Maxime Louis
Nicolas Warnet
maxime.louis@telecomnancy.eu
nicolas.warnet@telecomnancy.eu

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

LORIA
615 Rue du Jardin botanique
54506 Vandœuvre-lès-Nancy
03 83 59 20 00

Encadrants : Boniface Yann
Dutech Alain
Jacquey Evelyne

Remerciements

Nous adressons nos remerciements à nos encadrants Madame Jacquey Evelyne, Monsieur Boniface Yann et Monsieur Dutech Alain pour leur accueil au sein de leur équipe de recherche et leur disponibilité tout au long de notre projet.

Nous remercions tout particulièrement Monsieur Matthieu Nicolas pour son aide précieuse sur la structure de la PLM et ses liens avec la webPLM.

Table des matières

Remerciements **iv**

Table des matières **v**

1 Introduction **1**

2 État de l'art **2**

3 Légitimité du sujet **4**

4 Réflexion sur la teneur de l'environnement **5**

4.1 De la liberté du Buggle 5

4.2 Exploration des groupes d'idées exprimées par le langage 5

4.2.1 Déplacement physique 5

4.2.2 Descriptions statiques, relatives et comparatives 6

4.2.3 Descriptions d'émotions et de sensations 6

5 Aspects techniques **7**

5.1 La PLM - client Java 7

5.1.1 Infrastructure globale 7

5.1.2 Structure d'un exercice 8

5.2 La webPLM 8

5.2.1 Infrastructure 9

5.2.2 Fonctionnalités pré-existantes 9

5.2.3 Ajouts de fonctionnalités 9

5.2.4 Une base de données de descriptions 9

5.2.5 Installation et mise en oeuvre de l'application 10

6 Résultats **11**

7 Discussion globale **12**

8 Conclusion **13**

Bibliographie / Webographie **14**

Liste des illustrations **15**

Annexes **17**

A Visuels de l'application **17**

1 Introduction

Les équipes Maia et Cortex s'intéressent aux mécanismes de la cognition à des niveaux comportementaux et neuronaux et plus particulièrement à comment un agent cognitif (un robot par exemple) perçoit et représente son environnement et notamment à comment cet agent parvient à apprendre cette représentation.

Le langage naturel, c'est-à-dire le langage couramment employé, peut être un des supports de représentation, et les équipes Maia et Cortex étudient donc l'évolution de celui-ci. À l'aide du modèle d'apprentissage développé par Xavier Hinaut [1] qui s'appuie sur un réseau de neurones récurrent, il est possible de représenter l'évolution du langage par un modèle proche de celui de la logique des prédicats.

Ce modèle est ensuite utilisé pour associer des commandes de scripts avec du langage naturel. C'est ici que notre travail intervient. Afin de disposer d'un jeu d'essais suffisamment important, nous avons tâché de mettre en place, en s'appuyant sur la webPLM [4], une interface web permettant à des utilisateurs de créer leurs propres séquences vidéos, de les commenter, et de commenter celles des autres utilisateurs.

L'objectif de ce projet interdisciplinaire ou de recherche est, au-delà du développement technique, de découvrir les méthodes propres à la recherche, d'être capable de faire un état de l'art et de proposer une avancée vers la solution au problème suggéré.

Notre projet est disponible librement à l'adresse : *https ://github.com/nicolarexia/PIDR*.

2 État de l'art

Pour pouvoir apprendre à un robot le langage naturel, il est nécessaire de comprendre les fonctions du langage humain. Une de ces principales fonctions est de permettre les échanges et la coordination des actions entre humains.

La première approche, la plus naïve, est celle du dictionnaire : à chaque mot est associée une action ou une conséquence. Cependant cette approche à ses limites : elle ne permet pas l'apprentissage dynamique par un robot. Si les mots employés lui sont inconnus, il ne parviendra pas à interpréter la phrase, ce qui est l'objectif de l'apprentissage.

La seconde approche est d'observer la structure de la phrase et des éléments transmettant effectivement l'information. Il s'agit de trouver un lien entre les mots de contenu (c'est-à-dire les mots sémantiques) et leurs rôles dans le sens de la phrase. Le modèle proposé par Xavier Hinaut [1] permet d'analyser une phrase pour en extraire les sujets, verbes, objets, etc. Il est alors possible d'exprimer la phrase d'origine sous forme de prédicats avec entre autres un agent (celui qui effectue l'action) et un objet (celui qui subit l'action). Pour résoudre le problème d'attribution des rôles, une hypothèse est faite : *Le lien entre une phrase donnée et sons sens s'appuie sur le motif ordonné des mots et en particulier sur le motif des mots de fonction et des marqueurs.* L'intérêt de s'appuyer sur les mots de fonctions est qu'ils sont peu nombreux par rapport aux mots sémantiques. Ainsi, même si la sémantique des mots est inconnue du modèle, celui est capable de déterminer les liens interne de la phrase et sa structure.

La solution retenue, et celle qui est employée dans le projet auquel nous participons, est la modélisation du langage par un réseau de neurones. Une telle modélisation est proposée dans la thèse de Xavier Hinaut [1]. Ce réseau de neurones permet la décomposition de la phrase en prédicats.

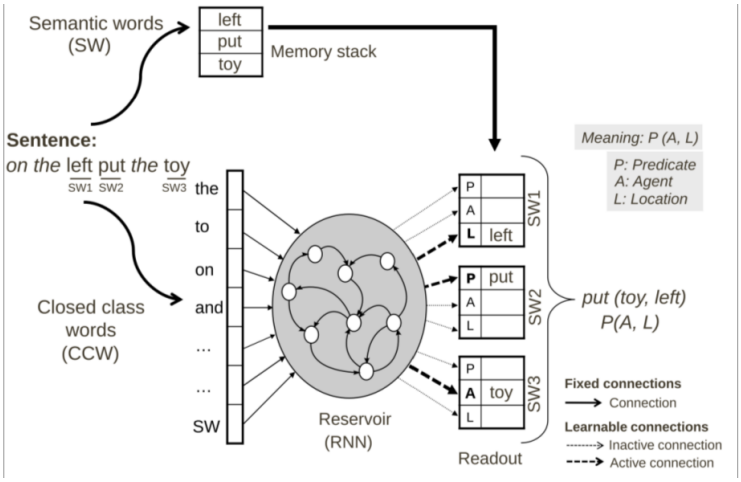

FIGURE 2.1 – Modèlisation du réseau de neurones par Xavier Hinaut

La figure 2.1 est une représentation de ce modèle. Au centre se trouve le réseau de neurones. Le

tableau à gauche contient les informations données en entrée. Ce sont les mots sémantiques de la phrase à analyser ou encore tous les mots de la phrase qui ne sont ni des mots de fonctions ni des marqueurs. Ces mots sémantiques sont associés à un chiffre puis transmis au réseau avec un certain poids permettant de fixer à quel point l'entrée extérieure influe sur le réseau. A la sortie du réseau, tous les mots sémantiques sont répertoriés et à chaque mot est attribuée sa fonction dans la phrase (agent, prédicat, objet...).

3 Légitimité du sujet

Le projet présenté dans se rapport est un des rares projets d'application du deep learning à l'apprentissage du langage humain français. Lors de nos recherches préalables, nous nous sommes vite rendus compte que la présence de corpus français traitant de ce sujet était assez limitée. Ainsi, la mise en oeuvre d'un tel projet nous semble légitime puisqu'elle peut déboucher sur de nouveaux axes de recherche, découlant d'informations que nous obtiendrions en sortie de notre réseau de neurones après lui avoir soumis suffisamment d'informations en entrée pour qu'il puisse nous fournir des résultats cohérents.

Afin de fournir au mieux ce corpus, nous avons fait le choix créer des séquences vidéos. Ces séquences pourront être commentées par un individu. Le résultat de la description sera ensuite donné en entrée du réseau de neurones. Une séquence vidéo sera donc une suite de fonctions informatiques agissant sur un avatar.

En outre, nous disposons d'un outil puissant pour déployer une telle application. Utilisée chaque année par des centaines d'étudiants, il s'agit de la PLM, initialement développée par Martin Quinson (Loria)

En modifiant certains principes de la PLM, nous souhaitons aboutir à un environnement qui permettra à l'utilisateur de faire se déplacer un avatar grâce à une série d'instructions. Mais comment permettre un maximum de possibilités dans cet environnement ?

4 Réflexion sur la teneur de l'environnement

Une phase importante du déroulement de notre projet a été la réflexion sur les possibilités que nous pouvions offrir aux utilisateurs de notre application. En effet, il était nécessaire que l'utilisateur dispose d'une certaine liberté au niveau des actions réalisables par son avatar : le Buggle. Plus encore, nous avons dû nous focaliser sur les différents aspects du langage humain, les différents domaines abordés lors de l'utilisation de celui-ci, les grands groupes d'idées qu'il permet d'exprimer.

4.1 De la liberté du Buggle

Après avoir fait l'inventaire des différentes actions réalisables par le Buggle grâce aux fonctionnalités de base de la PLM, nous avons déduit que les actions suivantes étaient pertinentes pour l'utilisation que nous souhaitions en faire : déplacement en avant et en arrière, virage à droite et à gauche, saisie d'un objet, marquage de l'environnement.

4.2 Exploration des groupes d'idées exprimées par le langage

Notre but étant d'obtenir des informations d'entrée hétéroclites pour notre réseau de neurones, il était primordial que nous permettions à l'utilisateur d'exprimer des idées de nature précise lorsqu'il décrit la séquence d'actions de son avatar à l'aide du langage humain. C'est pourquoi nous avons établi une liste des "champs" du langage, à savoir ce qu'il permet "matériellement" de décrire.

4.2.1 Déplacement physique

Le groupe d'idées le plus intuitif est d'ordre physique : il s'agit du déplacement du Buggle dans l'espace. Nous avons pris en compte les différents déplacements possibles (gauche, droite, bas, haut, diagonale ...) et avons déduit qu'il était possible de décrire la mouvance de l'avatar de diverses manières. En effet, bien que l'on puisse dire de façon élémentaire que l'avatar se déplace d'une case par la gauche, puis d'une par le haut, puis d'une par la droite et enfin d'une par le bas, il est également possible de décrire son déplacement à l'aide d'une figure géométrique : "le Buggle s'est déplacé selon un carré". Cette réflexion nous a également permis d'aboutir au fait que notre langage propose certaines fonctions de "compression" : au lieu de dire que l'avatar se déplace une fois à gauche, puis une fois à gauche, il est possible d'émettre l'assertion suivante :

"l'avatar se déplace deux fois à gauche". La compression du langage est donc établie à l'aide du comptage ordinal.

4.2.2 Descriptions statiques, relatives et comparatives

La puissance du langage humain permet de décrire une situation statique de beaucoup de manières différentes, allant de la plus vague à la plus précise. En effet, alors qu'on pourra dire "le Buggle se trouve près d'un mur", il est aussi possible d'être plus précis en statuant "le Buggle se trouve à une case du mur supérieur". Est alors introduite la notion de placement relatif. La description d'un objet se fait à partir de la description d'un autre objet. Sans même fournir d'information sur la substance de ces objets, on obtient des informations sur l'ensemble composé de ces deux objets. Des exemples d'outils linguistiques permettant de décrire de façon relative plusieurs objets sont par exemple "tourne autour de", "est près de", "au dessus de" etc. Enfin, il est possible d'induire la notion de comparaison en comparant deux objets sous la forme [A, opérateur de comparaison, B]. Par exemple, dire que le Buggle est plus haut que le mur inférieur serait une bonne information pour une entrée dans notre réseau de neurones. De façon analogue, on peut utiliser le superlatif : "le buggle est le plus mobile de tous les éléments de cette carte" serait une description correcte.

4.2.3 Descriptions d'émotions et de sensations

Il est très difficile d'aborder la notion d'émotion ("hot cognition" propre à l'humain) lorsqu'il s'agit de l'appliquer à un programme informatique ("cold cognition" propre aux machines). Cependant, il est possible de modéliser ces émotions à l'aide d'outils abstraits. Par exemple, la notion de couleur, plutôt abstraite, permet un large champ d'applications. Elle peut en effet symboliser les notions de température (c'est pourquoi on parle de couleur chaude et de couleur froide) mais peut aussi être utilisée pour décrire des émotions si on mappe efficacement chaque couleur à une émotion. Par exemple, une description utilisant cette notion pourrait être : "La couleur rouge représente la joie. Le buggle est joyeux car sa position correspond à celle d'une case rouge." Pourvu que l'on modélise correctement la correspondance entre couleur et une notion abstraite telle que l'émotion, il sera aisé de décrire des situations normalement inaccessibles à la compréhension par la machine.

5 Aspects techniques

Nous avons fait le choix d'utiliser la PLM comme support pour notre projet. La PLM, pour Programmer's Learning Machine [2], est une plateforme libre permettant l'apprentissage des bases de la programmation. Développée à l'origine sous la forme d'une application Java, elle est désormais disponible via une interface web : la webPLM [4].

L'utilisation de la PLM et de la webPLM nous permet d'ajouter du contenu à la PLM pour créer des séquences vidéos et d'exploiter l'implémentation web déjà existante de la webPLM. L'exportation vers une interface web et l'ajout de contenu sont plus faciles d'accès que sur une structure comme Scratch ou Blockly.

5.1 La PLM - client Java

La webPLM s'appuie entièrement sur le client Java de la PLM. Tout le contenu visualisable en ligne est généré à partir d'un fichier source .jar contenant le client Java.

Pour ajouter notre contenu, il a donc fallu en premier lieu modifier la PLM. Nous avons récupéré le code source de la PLM à partir de son dépôt public [3]. Une documentation est disponible. Malheureusement, celle-ci n'est plus maintenue à jour à cause des nombreuses transformations effectuées en ce moment sur l'infrastructure de la PLM. Pour pallier le manque d'informations, Matthieu Nicolas, ingénieur en charge de la PLM, a accepté de nous rencontrer afin de nous aiguiller et de nous donner quelques points d'entrée.

5.1.1 Infrastructure globale

L'ajout d'un exercice à la PLM doit respecter la structure déjà existante. Chaque nouvelle rubrique (dans notre cas la partie "description") doit être ajoutée au dossier "src" et être dans un package de la forme "lessons.maNouvelleRubrique". Ce package doit contenir une classe Main, indiquant quels sont les exercices liés à cette leçon. Il contient également des fichiers html permettant l'affichage d'informations et de descriptions à l'utilisateur.

Une fois notre nouvelle leçon mise en place, la création d'exercices se fait dans le package "lessons.description.exercice". C'est dans ce package que sont déclarés chaque exercice de la leçon ainsi que leurs descriptions.

Les liens entre les différentes leçons et leurs exercices sont gérés dans le package "plm.core.model" et plus particulièrement dans la classe "Game" qui répertorie la liste des leçons à afficher à l'utilisateur.

5.1.2 Structure d'un exercice

Chaque exercice implémente la classe "ExerciseTemplated". Il est défini par un jeu et par une leçon. Différents univers sont déjà disponibles comme le BuggleWorld ou HanoiWorld. Nous avons choisi de réutiliser le monde des Buggles car il est le plus approprié pour la création de séquences vidéos. Il propose de déplacer un avatar, le Buggle, sur une carte quadrillée.

Ce monde propose aussi différentes commandes comprises par le Buggle comme : avancer, reculer, tourner à droite et à gauche, prendre un objet, poser un objet, colorer le sol en différentes couleurs. Ce sont ces fonctions que nous réutilisons afin de créer des séquences vidéos.

Nous avons défini nos cartes à partir d'un fichier ".map". Chaque case de notre carte y est définie : sa couleur, si elle contient ou non un biscuit, les murs qui l'entourent. Il est également possible de définir le nombre de Buggles présent sur la carte et leur apparence.

Les différents exercices proposés

Afin de répondre au mieux à notre problématique et pour favoriser des descriptions originales, nous avons mis en place 4 exercices.

1. Bac à sable (voir figure A.1, page 17) : cet exercice propose à l'utilisateur une carte vierge. Il peut y découvrir les commandes comprises par les Buggles. L'intérêt de cette carte pour notre projet est de permettre à l'utilisateur de décrire les trajectoires effectuées par le Buggle. Par exemple, si le Buggle a décrit un carré, l'utilisateur peut le décrire comme un simple carré, ou bien comme la succession de 4 virages à droite.
2. Pièce (voir figure A.3, page 17) : Afin de proposer plus d'éléments de description, cette carte contient des murs que le Buggle ne peut pas franchir. L'utilisateur peut ainsi introduire des notions de placement dans l'espace comme "à droite du mur".
3. Couleurs ! (voir figure A.5, page 18) : Dans cette carte, les couleurs arrivent ! L'utilisateur peut ici imaginer des descriptions traduisant la chaleur (le bleu et le rouge), l'extérieur (herbe et ciel) ou encore des émotions.
4. Une petit faim (voir figure A.7, page 18) : Pour ajouter une dernière dimension aux descriptions, cette carte contient des objets, de délicieux biscuits, que le Buggle peut prendre et déplacer. L'utilisateur peut donc ajouter à sa description des notions d'actions et encore une fois de placement.

5.2 La webPLM

La WebPLM est issue, selon ses auteurs, de la volonté de passer d'un client Java lourd à une application web plus légère et facilement adaptable. Pour ce faire, le client Java sous forme de fichiers .jar est "lu" et directement intégré à une page web, combinant ainsi le résultat de plusieurs années de travail sur la PLM avec un environnement plus propice aux modifications futures. Notre projet tire profit des propriétés de cet outil puisque ce dernier permet le déploiement sur un serveur d'une application accessible par tous, par le biais d'internet.

5.2.1 Infrastructure

Tout comme pour la PLM, nous avons dû greffer notre projet sur une base de code pré-existante. En ce qui concerne la WebPLM, nous devions donc nous adapter à plusieurs langages et à plusieurs technologies différentes :

1. AngularJS, framework javascript, utilisé du côté du client.
2. Scala, utilisé du côté du serveur.
3. JSON, format de données textuelles répandu sur le web
4. MongoDB, base de données non-relationnelle

5.2.2 Fonctionnalités pré-existantes

Lorsque nous avons récupéré le code source de la WebPLM, l'outil nous proposait déjà certaines fonctionnalités utiles pour notre projet. Ces fonctionnalités, principalement issues du client Java de la PLM, nous permettaient déjà notamment de coder directement dans la page web grâce à un IDE intégré, de compiler ce code, et de pouvoir suivre son exécution, le tout en ligne.

5.2.3 Ajouts de fonctionnalités

La principale fonctionnalité que nous avons dû rajouter a été la possibilité pour un utilisateur de décrire une scène à l'aide du langage parlé, cette fonctionnalité étant inexistante dans le client Java. Gérée entièrement par l'applicatif web, son implémentation consistait en la création d'une nouvelle zone de texte sur l'interface de la WebPLM. Ainsi, lorsque l'utilisateur crée une scène à partir d'une séquence d'instructions qui lui est propre, il peut joindre une description textuelle en français des actions qu'il observe à l'exécution. Un simple clic sur un lien permet de déclancher la suite du traitement. L'étape suivante consistait à permettre la transition des données à partir du client, jusqu'au serveur. Pour réaliser cet acheminement, nous avons dû étudier la structure de la WebPLM afin de soumettre aux fonctions de transmission les formats de données corrects. Une fois les données arrivées sur le serveur, il ne restait plus qu'à les insérer en base de données.

5.2.4 Une base de données de descriptions

Un choix intuitif pour la base de données a été de s'orienter vers un système de gestion de base de données orientée documents, à savoir MongoDB. En effet, puisque le format des données est en JSON, il sera très facile pour les administrateurs de récupérer l'ensemble des données saisies par les utilisateurs. De même, il sera aisé par la suite de parser ces données afin de les soumettre au réseau de neurones en charge de l'analyse de la correspondance entre langage parlé et code prédicatif. La légèreté et la portabilité de la technologie MongoDB semblent justifier son utilisation dans le cadre d'un tel projet. Une base de données relationnelle classique aurait notamment pu poser des problèmes de performance si la taille des données venait à devenir trop élevée. Il est bon de garder en mémoire qu'à chaque insertion dans la base de données, une portion de code fait partie des données, ce qui peut, à terme, mener à l'obtention d'une base de données importante, qui se doit d'être maintenue. La gestion de MongoDB sur la webPLM a été assurée par un greffon au framework Play : ReactiveMongo.

5.2.5 Installation et mise en oeuvre de l'application

La procédure d'installation de l'environnement webPLM appliquée à notre projet de recherche est détaillée dans ce paragraphe.

1. D'abord, extraire l'archive .zip sur la machine censée faire office de serveur. Le framework scala "Play" doit être préalablement installé.
2. Dans un terminal, exécuter la commande suivante : *./activator start*. Cette commande a pour effet de déployer la partie serveur de l'application et de la rendre accessible sur un port particulier (ici le port 9000)
3. S'assurer de la présence d'une instance de mongodb sur le serveur. Si ce n'est pas le cas, l'installer, et démarrer le service associé à l'aide de la commande *service mongodb start*
4. L'application est utilisable ! Pour décrire une scène, remplir le champ situé en bas à droite de la page et cliquer sur "link code to description"
5. Pour récupérer les données des utilisateurs via l'interface MongoDB, lancer le shell mongo à l'aide de la commande *mongo* puis sélectionner la base de données *dbPLMDescription* à l'aide de la commande *use dbPLMDescription*. Les données recherchées se trouvent alors dans la collection "codeToDescription", affichables à l'aide de la commande suivante : *db.codeToDescription.find()*.

6 Résultats

Finalement, nous obtenons une application qui bénéficie à la fois de la base solide du projet PLM et des nouvelles fonctionnalités que nous avons implantées. Le tout est fonctionnel et fournit une interface de programmation simple (accès aux données via MongoDB) pourvu qu'une instance de MongoDB soit installée sur le serveur sur lequel notre application sera déployée.

Un exemple de données récupérables après une session d'utilisation de la PLM modifiée se trouve en annexe, figure A.9 à la page 18.

L'intérêt majeur de notre implémentation est sa compatibilité avec tous les exercices de la webPLM. Il est ainsi possible d'obtenir des descriptions de tous les exercices de la webPLM, comme la leçon Welcome contenant des exercices graphiques ou la leçon Turtle Art. De plus, la webPLM étant utilisée chaque année par les étudiants de Télécom Nancy pour apprendre la programmation, il est possible d'imaginer une grande quantité de données recueillies.

7 Discussion globale

Une idée d'amélioration serait de modifier à nouveau la PLM afin de proposer un éditeur de cartes grâce à laquelle les utilisateurs pourraient eux-même créer leur exercice, afin de les faire commenter par d'autres utilisateurs et de fournir encore un plus gros volume de données à notre base de données. Malheureusement, nous n'avons pu inclure une telle fonctionnalité, techniquement plutôt difficile à déployer. Les utilisateurs devront donc pour l'instant se contenter de créer des séquences à l'infini avec les cartes que nous proposons (et s'ils veulent ajouter d'autres cartes, il faudra qu'ils modifient le code source ...)

A titre personnel, ce projet nous a permis d'appréhender pour la première fois un logiciel de taille conséquente. Parvenir à comprendre le fonctionnement global de la PLM et de la webPLM a été pour nous un véritable défi. Nous avons pu voir comment est construit un tel code, quelles sont les bonnes pratiques et comment rendre ce code modifiable par tous.

La principale amélioration que nous pourrions réaliser serait de simplement proposer davantage de fonctionnalités dans l'environnement même du Buggle. Par exemple, afin de permettre des descriptions encore plus hétéroclites, nous pourrions créer un objet "sonore" sur la carte, qui pourrait être paramétrisé afin de faire varier la nature du son qu'il produirait. Ainsi, l'utilisateur pourrait donner des informations sur la hauteur du son, son gain etc.

8 Conclusion

Notre projet, bien qu'inachevé dans sa globalité (nous n'avons pas eu l'occasion de soumettre de paquets de données à un éventuel réseau de neurones) fournit tout de même un support stable de création de données d'entrée pour celui-ci. En effet, puisque nous l'avons développée de manière à ce qu'elle puisse être utilisable par n'importe qui dans le monde via internet, notre application a un potentiel infini en terme de génération de couples de données <code, description>. Lorsqu'un réseau de neurones travaillera à partir de ces couples de données, pourra-t-on confirmer son efficacité ou nous heurterons-nous à des problèmes liés à la substance même des informations saisies par les utilisateurs ?

Bibliographie / Webographie

[1] Xavier Hinaut. *Réseau de neurones récurrent pour le traitement de séquences abstraites et de structures grammaticales, avec une application aux interactions homme-robot.* PhD thesis, Université Claude Bernard Lyon 1, 2013. 1, 2

[2] Martin Quinson. Programmer's learning machine. http ://-people.irisa.fr/Martin.Quinson/Teaching/PLM/. 7

[3] Martin Quinson, Matthieu Nicolas, and Gerald Oster. Programmer's learning machine. https ://github.com/BuggleInc/PLM. 7

[4] Martin Quinson, Matthieu Nicolas, and Gerald Oster. Web interface of the programmer's learning machine. https ://github.com/BuggleInc/webPLM. 1, 7

Liste des illustrations

2.1 Modèlisation du réseau de neurones par Xavier Hinaut 2

A.1 Carte "Bac à sable" avant modification 17
A.2 Carte "Bac à sable" après modification 17
A.3 Carte "Pièce" avant modification 17
A.4 Carte "Pièce" après modification 17
A.5 Carte "Couleurs !" avant modification 18
A.6 Carte "Couleurs !" après modification 18
A.7 Carte "Une petite faim" avant modification 18
A.8 Carte "Une petite faim" après modification 18
A.9 Exemple de données récupérables via MongoDB 18
A.10 Visualisation de notre interface 19

Annexes

A Visuels de l'application

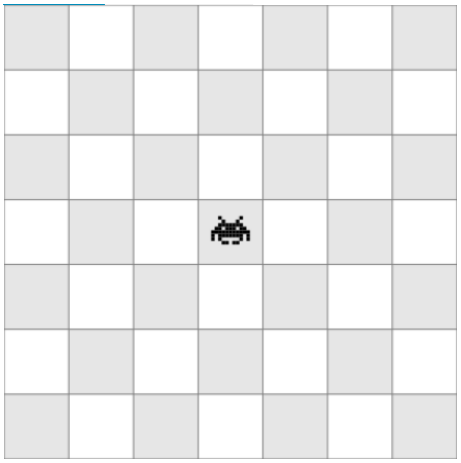

FIGURE A.1 – Carte "Bac à sable" avant modification

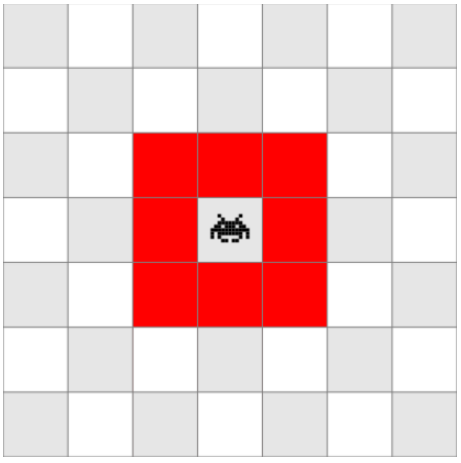

FIGURE A.2 – Carte "Bac à sable" après modification

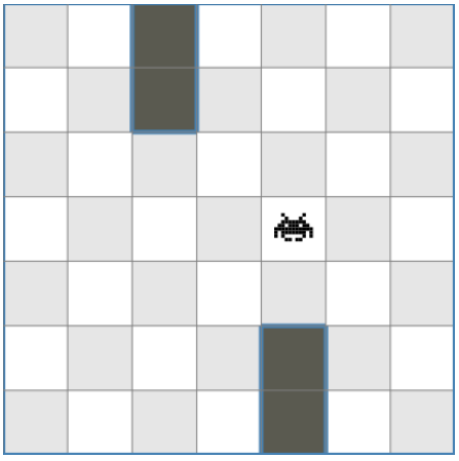

FIGURE A.3 – Carte "Pièce" avant modification

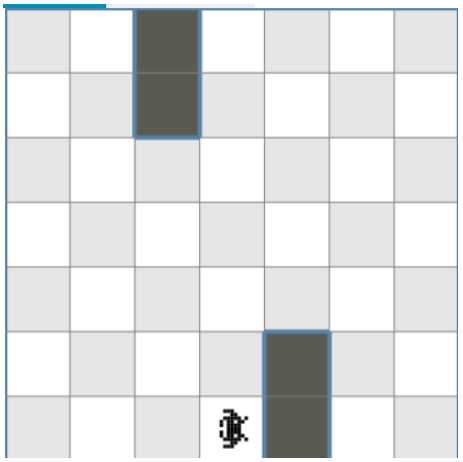

FIGURE A.4 – Carte "Pièce" après modification

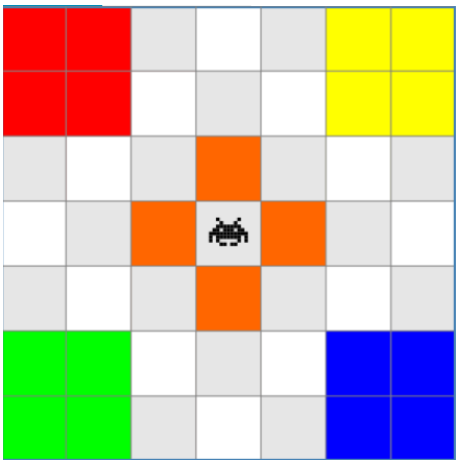

FIGURE A.5 – Carte "Couleurs !" avant modification

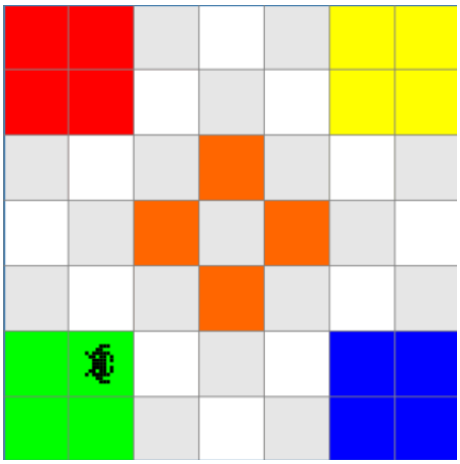

FIGURE A.6 – Carte "Couleurs !" après modification

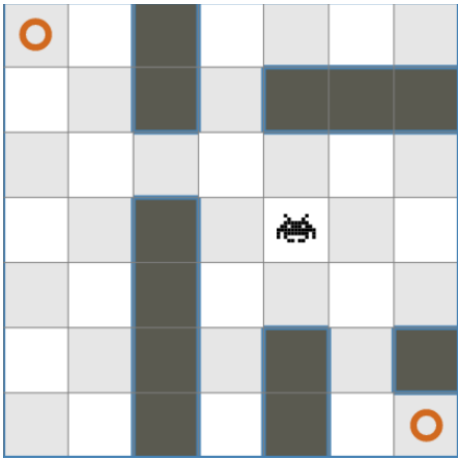

FIGURE A.7 – Carte "Une petite faim" avant modification

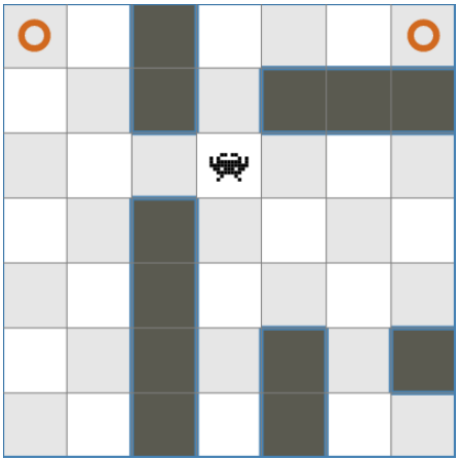

FIGURE A.8 – Carte "Une petite faim" après modification

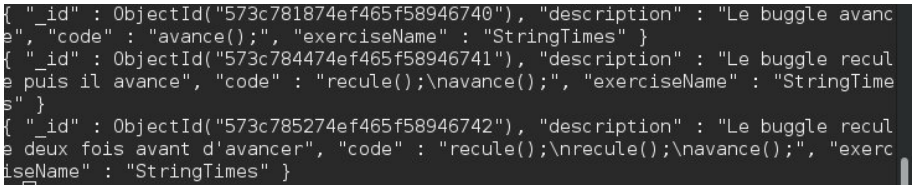

FIGURE A.9 – Exemple de données récupérables via MongoDB

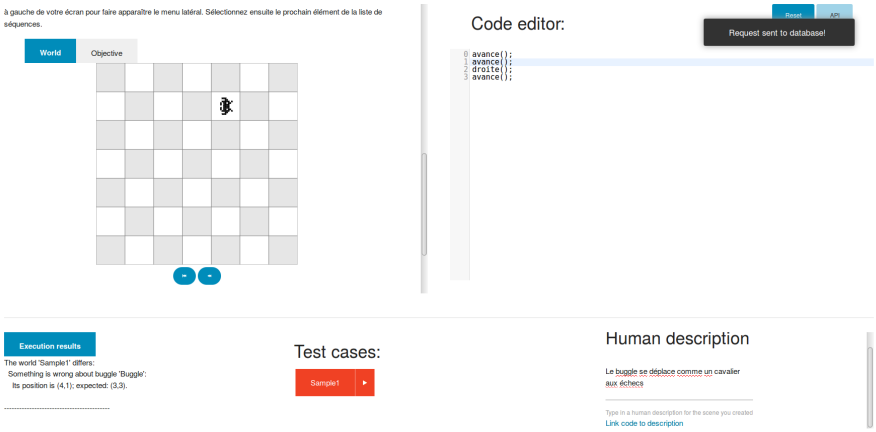

FIGURE A.10 – Visualisation de notre interface