Interfacer un modèle d'apprentissage de langage avec des vidéos issues de langage de programmation à boîtes

Graziella HUSSON
TELECOM Nancy, 2A
Ingénierie Appliquée aux Masses de Données
graziella.husson@telecomnancy.eu

Joris VIGNERON
TELECOM Nancy, 2A
Logiciels Embarqués
joris.vigneron@telecomnancy.eu

Encadrants
Yann BONIFACE
Alain DUTECH
Evelyne JACQUEY

***Résumé*—L'équipe Biscuit s'intéresse aux mécanismes de la cognition à des niveaux comportementaux et neuronaux. Et plus particulièrement à la façon dont un agent cognitif perçoit et représente son environnement, et sur les processus d'apprentissage de ces représentations.**

L'objectif est ici de mettre en place un site offrant la possiblité de collecter un corpus, suffisament conséquent pour permettre d'éprouver le réseau de neurones dont la tâche est d'apprendre à une intelligence artificielle à commander un robot à partir du langage naturel.

La réalisation de ce projet nous a poussé, tout d'abord, à étudier les différents langages de programmation à boîte dans le but de déterminer lequel pourrait convenir pour ce que nous souhaitons développer. Une fois notre choix fait, il a fallu approfondir les technologies employées par la plateforme d'aprentissage pour être en mesure de rajouter nos propres fonctionnalités.

***Mots-clés*—Réseau de neurones - Apprentissage langage naturel - Corpus - Web - PLM - WebPLM - Langage de programmation à boîte**

I. Introduction

L'équipe Biscuit est une équipe de recherche qui s'intéresse aux mécanismes cognitifs. Notamment aux niveaux du comportement et des neurones et plus particulièrement à la façon dont un robot ou tout autre agent cognitif serait en mesure de percevoir et représenter son environnement. L'autre aspect observé est la manière dont l'agent cognitif apprend cette représentation et utilise ces connaissances.

Cette équipe s'intéresse, dans le cadre d'une collaboration avec l'ATILF, au langage naturel et souhaite s'en servir comme support de représentation pour son agent cognitif. Pour cela, le modèle d'aprentissage développé par Xavier Hinaut [1], qui s'appuie sur un réseau de neurones récurrent, est utilisé. Ainsi il est possible de représenter l'évolution du langage par un modèle proche de celui de la logique des prédicats.

L'objectif du Projet Interdisciplinaire ou de Découverte de la Recherche au-delà d'un développement technique sur un cas concret, est de découvrir les méthodes propres de la recherche ainsi que toutes les étapes menant à son aboutissement : la réalisation d'un état de l'art, la proposition des avancées, l'évaluation des résultats et l'aboutissement vers une solution au problème posé.

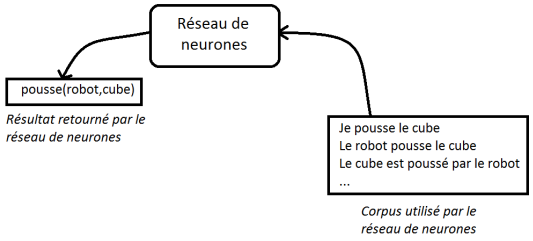

FIGURE 1. Schématisation du fonctionnement du réseau de neurones - transformation du corpus

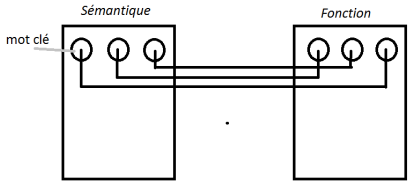

FIGURE 2. Schématisation du fonctionnement du réseau de neurones

II. Légitimité du sujet

L'objectif de notre Projet Interdisciplinaire ou de Découverte de la Recherche (PIDR) s'inscrit dans ce cadre. En effet, afin de disposer d'un corpus suffisamment important pour tester ce modèle d'apprentissage, il est nécessaire de développer une interface web en s'appuyant sur une technologie existante de plateforme d'apprentissage de la programmation. Ce choix est motivé par le fait que ces outils de découverte de la programmation utilisent généralement un langage à base de boîte facilement manipulable par des utilisateurs novices dans la programmation.
L'application web doit offrir la possibilité aux utilisateurs d'enregistrer leurs propres séquences vidéo, de les commenter et de récupérer ces commentaires pour les utiliser via le réseau de neurones.
L'intérêt est aussi que les utilisateurs ayant produit les animations ne les commentent pas afin de ne pas avoir une visualisation biaisée par ce qu'ils ont voulu faire plutôt que ce qu'ils ont réellement produit.
Pour réaliser cette application nous sommes passés par différentes étapes. Tout d'abord, la réalisation d'un certain nombre de recherches autour des langages de programmation à base de boîtes pour en choisir un sur lequel nous reposer. Nous avons le choix entre par exemple Snap, Scratch ou encore la PLM. Ensuite, nous a avons étudié en détails le fonctionnement de la plateforme de programmation qui utilise ce langage afin d'être en mesure de le modifier. Enfin, nous avons ajouté l'ensemble des fonctionnalités demandées.

III. Etat de l'art

A. Apprentissage du langage naturel

Afin d'apprendre le langage naturel à un agent cognitif, il est indispensable de comprendre les fonctions du langage humain.
La première approche est celle du dictionnaire : à chaque mot est associé une action. Cette approche à ses limites puisqu'elle ne permet pas un apprentissage dynamique : il faut à l'avance connaître la liste des mots connus et reconnaissables par l'agent cognitif. De plus, si un mot est inconnu ; il n'y a pas de manière simple de lui faire comprendre son sens, sans mettre à jour manuellement le dictionnaire.
Une seconde approche est d'observer la structure d'une phrase et des différents éléments de celle-ci porteurs d'informations. Il s'agit de trouver un lien entre les mots sémantiques (porteurs de sens) et leurs rôles dans la phrase. Le modèle proposé par Xavier Hinaut [1] permet d'analyser une phrase pour en extraire les sujets, verbes, 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).
La solution retenue, celle que l'équipe du LORIA emploie, est la modélisation sous forme d'un réseau de neurones (figure 3). Celle-là même proposée dans la thèse de Xavier Hinaut [1].

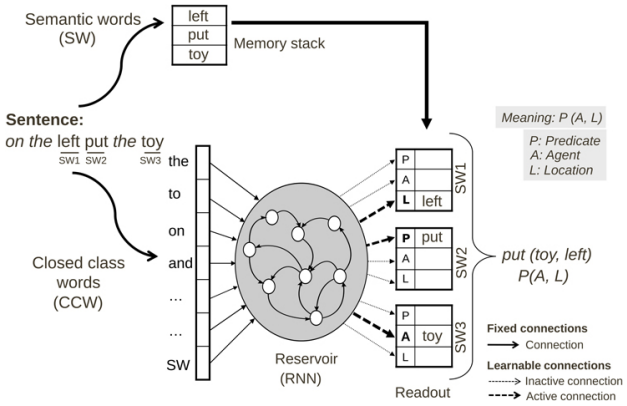

FIGURE 3. Modèlisation du réseau de neurones par Xavier Hinaut

Au centre de la figure 3 se trouve le réseau de neurones. Le tableau à gauche contient les mots sémantiques de la phrase à analyser.
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...).

B. langage à base de boîte

Afin de permettre à n'importe quel utilisateur de manipuler l'application nous avons utilisé un langage de programmation à base de boîtes. Il s'agit de langages majoritairements graphiques à base de boîtes ou de flèches. Chacun de ces éléments graphiques dispose d'une fonction au sein du langage : tester une variable, affecter une valeur, boucler sur une instruction, etc. Le point fort de ce type de langage est qu'il se rapproche du langage parlé d'une certaine manière et donc bien plus intuitif pour un utilisateur non familiarisé aux langages de programmation classiques.
La majorité de ces langages sont utilisés par des plateformes de programmation afin d'introduire le développement d'applications informatiques auprès d'usagers. Lors de nos recherches, nous avons pu en relever trois qui nous ont semblé adéquats pour la réalisation du projet.

Scratch : résultat de l'implémentation visuelle et technique du langage de programmation SmallTalk fondée sur Squeak. Il s'agit d'une plateforme dont l'objectif est l'apprentissage de l'informatique aux enfants. Le langage est entièrement basé sur un système de boîtes : chaque boîte représente une instruction classique de la programmation, l'objectif est d'associer les boîtes entre elles pour réaliser un code, certaines boîtes ne peuvent s'emboîter car cela entrainerait une erreur syntaxique ou sémantique.

Snap : combinaison de Scratch avec son interface graphique et son utilisation simple avec drag-n-drop et Scheme pour sa

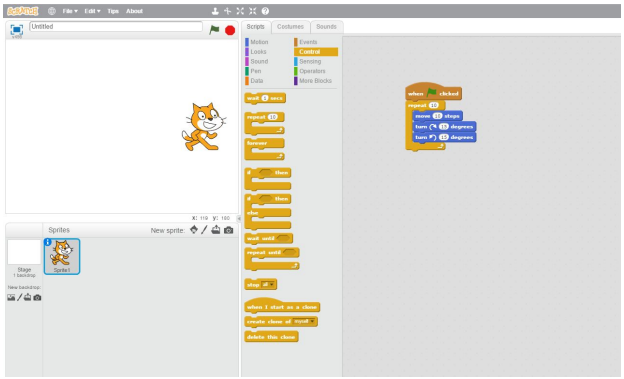

FIGURE 4. Interface de Scratch

base de programmation. Il s'agit donc également d'une autre plateforme d'apprentissage de la programmation basée sur un langage à base de boîte. L'interface est bien plus graphique que scratch et dispose d'un écran pour afficher le résultat de l'exécution.

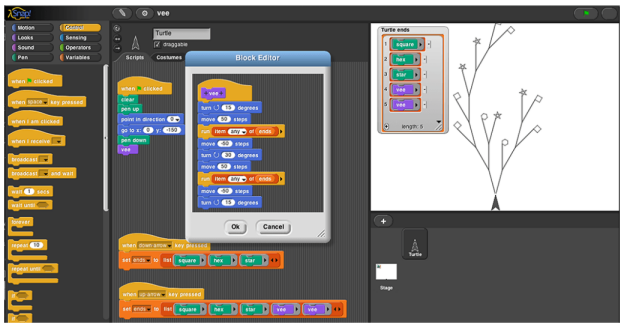

FIGURE 5. Interface de Snap

PLM : plateforme d'apprentissage de programmation développée par le LORIA et qui n'utilise pas un système de boîte mais de séquences d'instructions écrites en langage naturel : avance, recule, droite, gauche,... Elle dispose également d'un cadre d'exécution dans lequel on peut observer un petit personnage, un Buggle, évoluer dans son environnement. Différents exercices sont proposés au sein même de la PLM afin de travailler certains aspects en particulier. Chaque exercice a un objectif à atteindre, l'utilisateur doit donc produire le code nécessaire pour obtenir le résultat escompté.
Les exercices proposés peuvent être simplement de trier une liste,de résoudre un problème de type "tour de Hanoï" ou autre.

C. *Réseau de neurones*

Un réseau de neurones artificiels, ou d'apprentissage supervisé s'appuyant sur des exemples de comportement souhaité, est un ensemble d'algorithme dont la structure est inspirée de

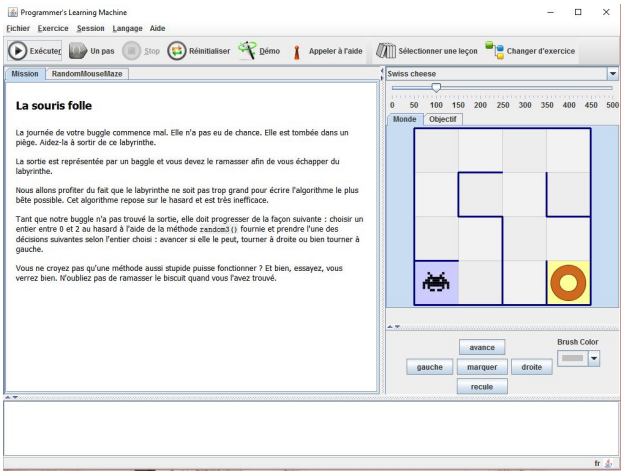

FIGURE 6. Interface de la PLM

celle du cerveau. Ils se basent généralement sur le principe d'apprentissage automatisé. Cela signifie que les algorithmes utilisés adaptent leur comportement et leur analyse au fur et à mesure qu'ils obtiennent des informations sur le sujet. Ainsi, plus les cas d'essais ce multiplient et plus les réponses des réseaux de neurones seront pertinentes. Ce genre d'algorithme est généralement utilisé pour doter des machines ou des ordinateurs de systèmes de perception de leur environnement, de moteurs de recherche ou encore d'aide aux diagnostics.

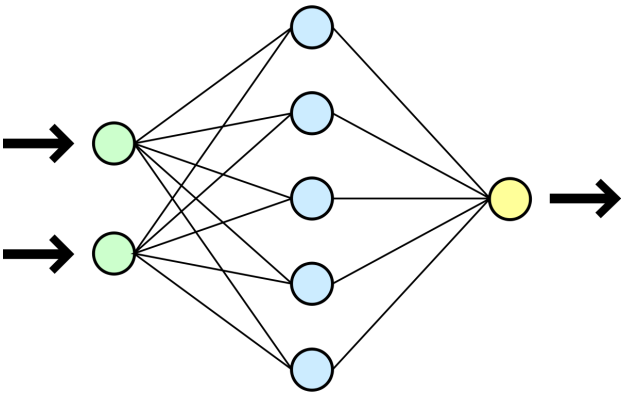

FIGURE 7. Vue simplifiée d'un réseau de neurone artificiel - tirée de wikipédia

IV. MÉTHODOLOGIES

Dans cette partie nous allons expliquer comment nous avons développé notre application. Il nous fallait mettre en place une application de type web qui permette à un utilisateur de coder une séquence d'instructions, d'enregistrer un rendu graphique de l'exécution du programme et de commenter les vidéos qui ont été réalisées par d'autres utilisateurs.
Afin de simplifier le travail, nous avons pris la décision, en accord avec l'équipe de recherches Biscuit, de nous baser sur un langage de programmation à base de boîtes. Ce genre de langage vise à permettre à des utilisateurs non aguerri à la

programmation de pouvoir coder des séquences d'instruction simples. En général, ces langages sont implémentés sur des plateformes qui sont elles-mêmes diffusés au public pour introduire la programmation aux non initiés. Nous devions donc choisir l'une de ces plateformes et nous en servir comme base de notre application.
Dans un premier temps, nous avons retenu trois d'entres elles, soit parce que nous les connaissions, soit parce que nos encadrants nous les avaient proposées. Il s'agit de Snap, Scratch et la PLM. Pour faire notre choix nous avions décidé de nous fixer quelques critères comme la facilité d'utilisation du langage pour les utilisateurs, l'obtention des sources de la plateforme et surtout la possibilité de la modifier afin de développer notre application. Après un certains temps à essayer chacune d'entre elles nous avons finalement décidé de choisir la PLM. Tout d'abord, parce qu'il s'agit d'un projet développé par des équipes de recherches du LORIA, ce qui nous permettait de pouvoir éventuellement contacter certains des professionnels qui avaient travaillé dessus. De plus, les sources étaient facilement accessibles via Github. Enfin, la PLM se basait sur un langage de programmation à base de boîte inspiré du langage courant et présentait une interface graphique affichant les résultats des exécutions. Une fois ce choix fait, nous avons pu nous pencher sur le développement de notre application web et ce en suivant un certain nombre d'étapes de réalisation.

A. Etude de la PLM et de la WebPLM

Une fois notre choix de plateforme de programmation fait, il nous fallait tout de même étudier l'achitecture générale de celle-ci. Il s'avère que l'application web que nous avons eu l'habitude d'utiliser n'était en fait qu'une surcouche web placée au-dessus d'un programme, lui écrit en Java. A partir de maintenant nous distinguerons donc les deux, d'un côté il y a la PLM et de l'autre la webPLM.
La PLM défini les règles de vérifications du code écrit par les utilisateurs, génèrent les messages d'erreurs et met à jour l'affichage de l'interface graphique. Elle dispose d'"exercices" définis qui sont utilisés pour apprendre la programmation.
La webPLM fonctionne pour chaque "leçon" proposée par la PLM. Une page d'accueil recueillait la liste de toutes les leçons possibles afin de permettre a l'utilisateur d'aller sur la page de l'exercice qui l'intéresse. Il pouvait alors écrire un code à droite et le tester. A chaque test, le programme vérifiait si l'objectif était ateint, et le cas échéant stoppait l'exercice pour annoncer sa réussite. La webPLM est issue, selon ses auteurs, de la volonté de passer d'un client Java lourd a une application web plus légère et facilement adaptable. Pour ce faire, le client Java sous forme de fichiers .jar est directement intégré à une page web, combinant ainsi le résultat de plusieurs années de travail sur la PLM. Elle est basée sur un ensemble de technologies telles que :

— AngularJS, framework javascript, utilisé du côté du client.
— Scala, utilisé du côté du serveur. Il s'agit d'un langage asynchrone intègrant les paradigmes de programmation orientée objet et de programmation fonctionnelle, avec un typage statique. Il concilie ainsi ces deux paradigmes habituellement opposés.
— ScalaPlay, framework web open source qui permet d'écrire rapidement des applications web en Java ou en Scala.
— Akka, un framework pour écrire des applications concurrentes, scalabes et robustes. Il s'agit de faire communiquer simplement différents processus par échange asynchrone de messages. Chaque acteur dispose d'une boîte à messages qui contient les messages que le monde extérieur lui a envoyé. Un acteur reçoit juste des messages, fait des traitements et envoie des messages.
— ReactiveMongo, utilisé pour gérer une base de données MongoDB orientée documents. Il s'agit d'un greffon au framework Play.

Toutes ces technologies travaillent ensemble afin de permettre à la webPLM de fonctionner.

B. Mise en place d'exercices de types "bac à sable"

Pour notre application, nous souhaitions que l'utilisateur soit libre de développer sa propre séquence d'instruction et ce sans aucun restriction. Le problème est que, dans la PLM, un certain nombre d'exercices ont déjà été développés et que chacun d'entre eux respecte la même règle : il faut qu'un objectif soit atteint à la fin de l'exécution du programme sinon la plateforme génère une erreur. Or nous souhaitions pouvoir mettre en place notre propre exercice dans lequel l'utilisateur serait libre de faire ce qu'il souhaite. C'est pourquoi nous avons dû passer par une première phase d'étude du code de la PLM afin de répondre à quelques questions que nous nous posions : tout d'abord comment créer un exercice, ensuite comment faire en sorte que cet exercice soit libre et ensuite qu'aucune erreur ne soit déclenchée au moment de l'exécution du programme de l'utilisateur.
Après discussion avec nos encadrants, nous avons appris qu'un autre groupe d'étudiants avait travaillé sur un projet similaire et notamment sur la PLM. Nous avons pu avoir accès à leur travail et ainsi étudier les solutions qu'ils avaient proposé. Puis nous les avons contacté afin de pouvoir échanger avec eux sur le sujet et qu'ils puissent nous expliquer clairement de quelle façon la PLM gérait les exercices.
Une fois nos connaissances acquises sur la PLM, il fut temps de nous mettre au développement de l'application. Nous nous sommes alors inspiré des exercices implémentés par les étudiants que nous avions rencontré qui étaient de type SandBox. Il y en avait disposant de murs empêchant le petit personnage, nommé Buggle, de passer, un autre avec des couleurs au sol exprimant les sentiments et les impressions, et enfin un dernier dans lequel était disposé des biscuits au sol que le buggle pouvait prendre et déplacer. Bien que nous l'ayons repris de leur travail, il nous a tout de même fallu revoir la structure de leur code qui ne respectait pas l'architecture globale de la PLM, nous empêchant alors d'avoir accès

directement aux exercices plutôt qu'à la leçon uniquement.
Pour réaliser les modifications nécessaires, il nous a fallu nous baser sur un certain nombre de règles de structuration imposées par l'architecture de la PLM. Chaque exercice implémente la classe "ExerciseTemplated". Il est défini par un jeu et par une leçon. Chaque exercice contient une liste de commandes disponibles, telles que "avance()", "recule()", "droite()", etc. pour les exercices du type BuggleWorld. Ce sont ces fonctions que nous réutiliserons afin de créer des séquences vidéos. Chaque exercice défini une "carte" à partir d'un fichier ".map". Chaque case de la carte y est définie : sa couleur, si elle contient ou non un biscuit et les murs qui l'entourent. Il est également possible de définir le nombre de Buggles présents sur la carte et leur apparence.
A présent nous disposons de carte permettant de coder librement. Cette partie terminée nous pouvions à présent nous concentrer sur la partie de l'application qui devait générer les vidéos. Cependant, cette partie allait devoir être réalisée sur la webPLM et non sur la PLM.

C. Création de séquences vidéo via la PLM

Comme expliqué précédemment, cette partie de développement va être réalisée sur la webPLM. Tout comme pour la PLM, une phase d'étude s'est imposée pour pouvoir développer nos différentes fonctionnalités. Dans cette partie il fallait nous interroger sur les points suivants : comment modifier l'interface graphique de la webPLM et de quelle manière nous pourrions enregistrer le rendu graphique que produit la webPLM.

Nous avons alors du étudier la webPLM et son fonctionnement interne. Le chercheur Nicolas Matthieu du LORIA, qui travaillait sur la webPLM, a pu nous apporté son aide dans cette étude. De plus, il nous a fourni différentes pistes à développer pour nous permettre de créer un système qui enregistrerait le rendu graphique que la PLM fournit au moment de l'exécution du code. Finalement la sauvegarde de séquences se fait grâce au canvas qui gère l'affichage du rendu. A chaque modification de l'affichage lors de l'exécution du code, nous sauvegardons une image de l'interface. Au final nous obtenons une successions d'images qui, mises bout à bout, forment une vidéo de l'exécution.

Une fois la séquence d'images obtenue il nous faut les sauvegarder. Nous avons donc mis en place une base de données orientée documents. Celle-ci contiendra plusieurs fichiers JSON (structurés) contenant le nom de l'exercice, la liste des images et le code associé. Ces informations, pour être stockées dans la base de données, doivent passer par le framework akka. Celui-ci utilise un système de messagerie afin de faire converser le javascript avec le scala qui se charge d'intéragir avec les bases de données. Ce fonctionnement est expliqué sur la figure 8.

Prenons un exemple pour bien comprendre comment cela est structuré. La page code.html souhaite insérer le code, le nom de l'exercice et la liste des images obtenues dans la base de données. Le nom de l'exercice et le code sont indispensables pour éviter les doublons dans la BD. En effet, nous souhaitons avoir une liste de commentaire pour une même animation, chose impossible si la BD contient plusieurs fois la même animation. Le controller de cette page, code-controller.js, va envoyer un message en utilisant le framework akka. Ce message comprendra en arguments ces informations et utilisera la commande "insert(element)" par exemple. Ce message sera reçu par le scala qui va exécuter la commande et envoyer les images vers notre base de données pour les stocker.

A son arrivée sur l'application, l'utilisateur arrive tout d'abord devant la page home (il s'agit de la page principale de l'application) dans laquelle il peut choisir l'un de nos "exercices". Une fois ce choix fait, il peut alors coder ce qu'il souhaite et même tester son code via le bouton "Executer". Une fois sûr de lui, il peut sauvegarder ce même code ainsi que la vidéo en appuyant sur le bouton "Enregistrer" qui va, par le biais du framework Akka, envoyer ces informations dans la base de données.

A présent notre application est capable de sauvegarder les images obtenues par l'exécution d'un code produit par un utilisateur dans l'un de nos exercices.

D. Enregistrement des commentaires

Cette partie de l'application doit permettre à l'utilisateur de commenter les vidéos qui ont été enregistrées auparavant. La page "commentaires" charge une liste de toutes les animations disponibles (elle va donc chercher tous les éléments de la base de données précédente). L'utilisateur peut alors cliquer sur l'une des animations afin de la commenter. Son commentaire sera en fait de la forme "La Buggle s'est déplacé en emportant un biscuit." par exemple. Une page s'ouvre alors afin de lui afficher la dite animation (qu'il peut rejouer, mettre sur pause, etc.) et de lui permettre de rentrer son commentaire. Une fois qu'il le sauve, nous allons insérer dans une deuxième BD le nom de l'exercice, le code associé et le commentaire de l'utilisateur. Nous allons donc mettre à jour la BD si un commentaire existe déjà afin encore une fois d'éviter les doublons.
Techniquement, nous nous reposons à nouveau sur le framework Akka qui va servir d'interface entre le Javascript qui va gérer l'affichage sur la page web et la base de données qui contient les informations. Il nous permet de récupérer la liste des animations dans un premier temps, puis la liste des images d'une vidéo en particulier dans un second temps. Afin d'obtenir une vidéo à partir de celles-ci, Javascript va les lire dans l'ordre et, au fur et à mesure, va remplacer l'image précédente dans le fichier HTML.
A présent nous disposons d'une application qui en plus de

FIGURE 8. Schéma présentant la structure que nous avons utilisé pour accèder aux bases de données

sauvegarder la vidéo d'un code peut enregistrer des commentaires.

E. Ajout de fonctionnalités additionnelles

Cette dernière partie concerne des fonctionnalités que nous avons ajoutées afin d'offrir davantages de services aux utilisateurs. Tout d'abord nous avons mis en place une page administrateur qui regroupe toutes les informations dont l'équipe Biscuit aura besoin : le nom de la vidéo, le code qui lui est associé et les commentaires. Ainsi ils n'auront qu'à se rendre sur cette page pour obtenir l'ensemble du corpus dont ils ont besoin. Pour cette fonctionnalité, nous utilisons les mêmes méthodes que pour récupérer une vidéo ou enregistrer les commentaires.
De plus, nous avons mis en place une page d'aide aux utilisateurs afin qu'ils puissent utiliser notre application facilement. Des vidéos explicatifs et de démonstration on été incorporées.

Pour résumé le fonctionnement de notre application nous pouvons dire que la PLM nous a permis de mettre en place des exercices de types SandBox offrant la possiblité aux utilisateurs de coder sans contraintes. Après compilation des sources Java, la PLM génère trois .jar qui sont utilisés par la webPLM comme base.

Lorsqu'il démarre l'application, l'utilisateur se retrouve devant la page home qui lui propose de choisir un exercice. Il peut alors coder une suite d'instructions, la tester et finalement l'enregistrer.
Il peut également se rendre sur la page de commentaires pour arriver devant un menu qui lui propose une liste de vidéo enregistrées dans la base et qu'il peut commenter. Pour ce faire, il peut regarder l'animation en question puis écrire et enregistrer un commentaire.
Finalement, notre application propose une page administrateur permettant aux membres de l'équipe Biscuit d'avoir rapidemment accès aux informations dont ils ont besoin comme la séquence de code et les commentaires qui lui sont associés. Nous avons également une page d'aide pour permettre aux utilisateurs de comprendre comment utiliser notre application.

La figure 9 résume l'organisation de notre application.

V. RÉSULTATS

Dans ce projet nous avons donc en place une interface web offrant la possibilité aux utilisateurs, par le biais d'un langage de programmation à base de boîtes, de réaliser de courtes séquences vidéos. Ces vidéos peuvent être commentées par d'autres utilisateurs pour réaliser un corpus. Celui-ci permettra par la suite à l'équipe Biscuit d'avancer dans leur recherches sur les mécanismes cognitifs.

En ce qui concerne les résultats obtenus, notre application réalise les tâches qui lui ont été confiées correctement. Elle propose des environnements de travail de type sandbox et

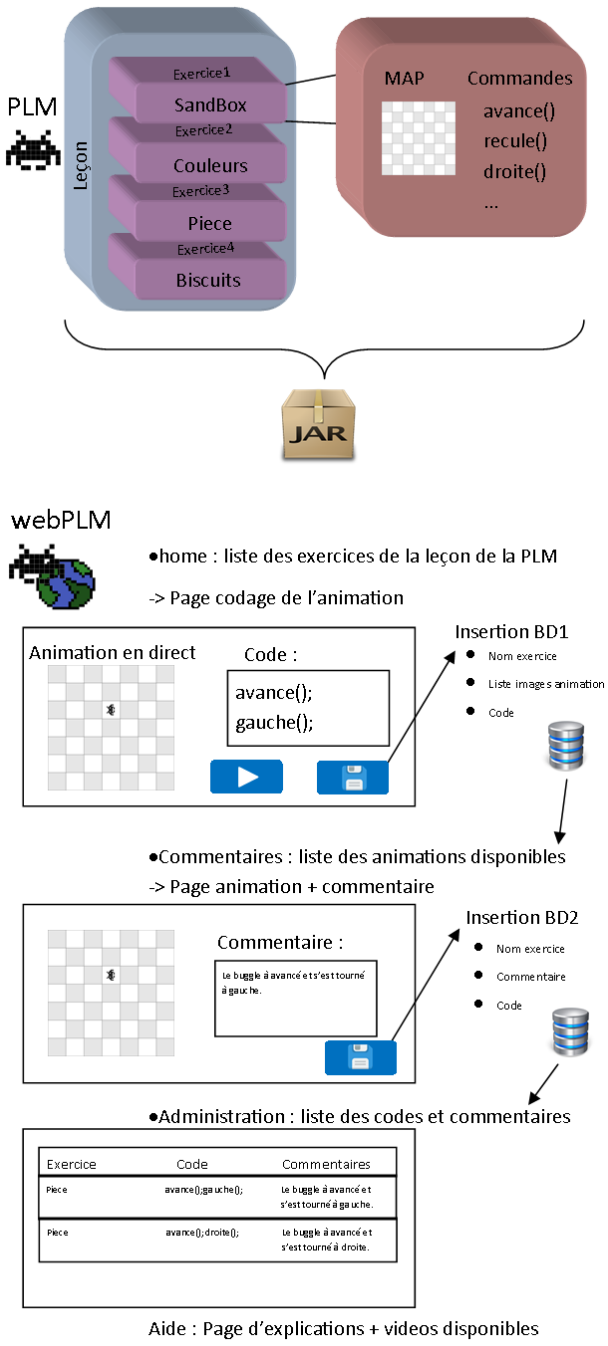

FIGURE 9. Schéma de nos réalisations

permet l'enregistrement de séquences vidéos. Elle permet de visualiser les dites vidéos et de les commenter. Enfin, les administrateurs peuvent accéder à une page web qui récapitule l'ensemble des vidéos avec leurs codes et leurs commentaires.

Finalement, l'application étant utilisable par tout type d'utilisateur, ce corpus ne sera pas biaisé. En effet, si seul les membres de l'équipe de recherche pouvaient réaliser les vidéos, les résultats seraient finalement moins intéressants pour eux.

VI. DISCUSSION GLOBALE

L'application présente toutefois des axes d'améliorations possibles. En effet, nous avons remarqué une petite limitation liée à la taille des messages utilisés par le framework akka. Une solution pourrait être de trouver une autre méthode de transmission vers la base de données. La solution que nous avons mise en place consiste à allouer une taille plus grande pour ce buffer (permettant ainsi de générer des animations assez longues), mais la limitation reste présente.
Une autre voie de développement pourrait être l'ajout de plus de possibilités dans la configuration de l'univers de type sandbox. En effet, actuellement l'utilisateur programme sur un environnement certes libre mais préconçu par nos soins. L'idéal serait effectivement de permettre aux utilisateurs de configurer leur environnement de travail en choisissant eux-mêmes les décors : nombre et emplacement des murs, des biscuits, etc.

Une autre limitation de notre solution est le fait que deux codes différents peuvent mener à la même visualisation. En effet, plusieurs des méthodes pouvant être utilisées comme "lèveBrosse()" et "baisseBrosse()" mène pas à une animation différente : le buggle ne bouge pas. Cela a pour conséquence d'avoir dans la base de données deux codes différents qui vont mener à la même animation et donc éventuellement aux même commentaires. Aucun commentaire ne pourra être fait sur ces instructions puisqu'elles ne sont pas visibles.

Lors de la réalisation de notre application nous avons rencontré un certain nombre de difficultés qu'il nous a fallu surmonter. Par exemple, l'utilisation et la modification de la PLM pour réaliser notre application s'est avérée plus difficile que nous le pensions de par sa structuration particulière. C'est pourquoi les personnes que nous avons contacté nous ont été d'une grande aide dans la compréhension du code de la PLM.
L'une des autres difficultés fut lors de la mise en place du code pour enregistrer les vidéos. En effet, il n'était pas évident au premier abord, de déterminer quelle partie de la webPLM il nous fallait modifier pour implémenter cette fonctionnalité.
La dernière difficulté fut rencontrée en fin de réalisation. En effet, lors de l'enregistrement d'une vidéo de taille trop importante, la socket de dialogue vers le serveur n'était pas en mesure, comme elle était configurée au départ, de contenir toutes les informations. Une socket est une interface de connexion entre deux acteurs dans un réseau . Au moment de sa définition, il est nécessaire de déterminer la taille du buffer de réception, dans notre cas cette taille était bien trop petite par rapport avec ce que nous souhaitions envoyer.Il nous a fallu beaucoup de temps pour trouver le paramètre à modifier afin d'augmenter la taille de cette socket.
Au vu des éléments exposés précédement il est possible de

discuter le choix de la PLM en tant que base de réalisation pour notre application. En effet, les petits soucis rencontrés sont liés à la PLM elle-même et au fait que nous ayons dû nous adapter à son mode de fonctionnement. Cependant, nous voulons souligner le fait que parmi les autres possibilités que nous avions, la PLM reste le meilleur choix en terme d'accessibilité et de documentation.

VII. Conclusion

Finalement nous pouvons dire que ce projet nous aura permis de découvrir l'univers de la recherche sur un cas concret. Nous avons notamment pu expérimenter les différentes phases d'un projet de recherche : la réalisation d'un état de l'art, la réflexion sur des solutions, la mise en oeuvre de celle-ci, etc. Nous avons également pu mettre en oeuvre un certain nombre de technologies que nous n'aurions peut être jamais utilisées lors de notre formation.
Vous trouverez sur Github notre code source libre d'accès : https://github.com/Graziella-Husson/PIDR-2017

Remerciements

Nous tenons à remercier nos encadrants Mme Jacquey Evelyne, M. Boniface Yann et M. Dutech Alain pour leur accueil au sein de leur équipe de recherche et leur disponibilité tout au long de notre projet.
Nous remercions également M. Matthieu Nicolas pour son aide sur la structure de la webPLM et M. Warnet Nicolas, membre du binôme ayant eu un sujet similaire l'année passée, pour son aide sur les modalités de modification de la PLM.

Références

[1] "Constructions grammaticales et interactions homme-robot", chapitre 7 de la thèse de Xavier Hinaut.

[2] X. Hinaut S. Wermter "An incremental approach to language acquisition : Thematic role assignment with echo state networks"

[3] X. Hinaut J. Twiefel M. Petit P. Dominey S. Wermter "A recurrent neural network for multiple language acquisition : Starting with english and french"

[4] X. Hinaut J. Twiefel M. Borghetti Soares P. Barros L. Mici S. Wermter "Humanoidly speaking - learning about the world and language with a humanoid friendly robot"

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

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

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

Glossaire

agent cognitif Les systèmes d'agents cognitifs sont fondés sur la coopération d'agents capables à eux seuls d'effectuer des opérations complexes. 1

ATILF Analyse et Traitement Informatique de la Langue Française. 1

BD Base de données. 5

corpus Un corpus est un ensemble de documents (textes, images, vidéos, etc.), regroupés dans une optique précise. Ici, dans le but de "nourrir" un réseau de neurones. Ce corpus sera formé de plusieurs phrases décrivant un code comme par exemple "Le robot a avancé". 1, 2, 6, 7

Github est un service web d'hébergement et de gestion de développement de logiciels de gestion de versions Git. Nous l'avons également utilisé pour stocker notre code. 4, 8

LORIA Laboratoire lorrain de recherche en informatique et ses applications. 2–5

PLM Programmer's Learning Machine: Outil interactif et graphique pour apprendre à coder. Développé par une équipe du LORIA. 2–8

prédicat Élément central de la phrase, autour duquel s'organise la fonction des autres éléments. 1, 2

webPLM Interface web de la PLM. Développé par une équipe du LORIA. 4–8

ANNEXES

Capture d'écran de l'éditeur de code :

FIGURE 10. Page permettant de coder une petite animation

Capture d'écran de la page permettant d'ajouter des commentaires. La partie gauche est une animation dynamique se jouant et pouvant être rejouée, mise en pause ou arrêtée.

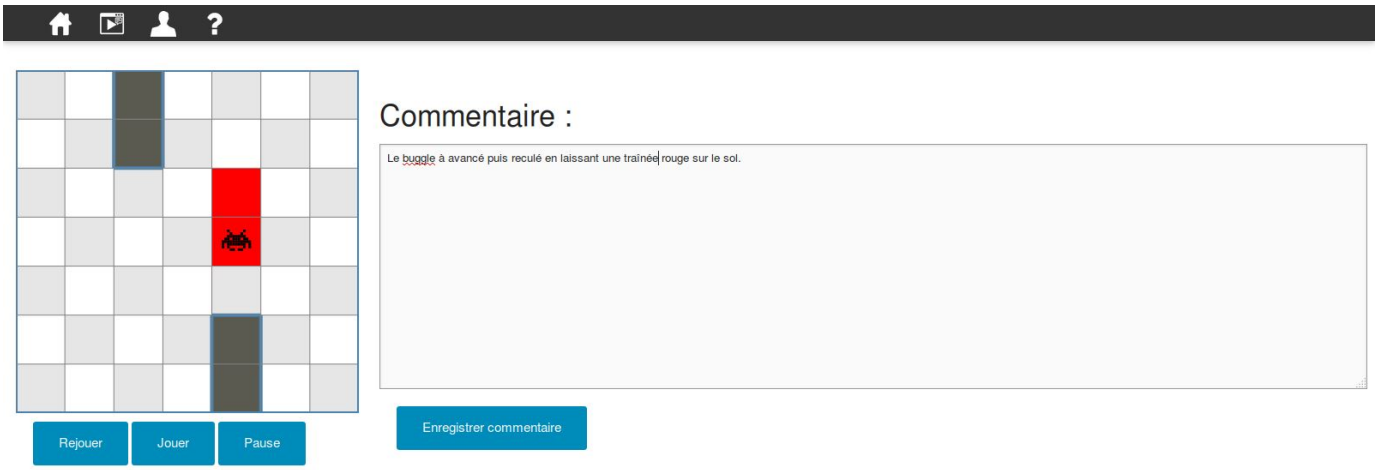

FIGURE 11. Page permettant de commenter une petite animation

Capture d'écran de la page d'administrateur permettant de visualiser les commentaires et les codes générés par tous les utilisateurs

Page Admin

Exercise Name	Code	Descriptions
Description.Piece	setCouleurBrosse(Color.RED); baisseBrosse(); avance(); recule(); leveBrosse();	Le sprite bouge et le sol devient rouge. ----- Le buggle à avancé puis reculé en laissant une trainée rouge sur le sol. -----

FIGURE 12. Page permettant de visualiser les commentaires et les codes