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 permettant de collecter un corpus, suffisament conséquent pour permettre d'éprouver le réseau de neurone permettant 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 par la suite 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 intéressée par les 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 en particulier au langage naturel, le langage couramment employé, 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.

II. 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 connaitre 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.
La seconde approche est d'observer la structure d'une phrase et des différents éléments de celle-ci porteurs d'informations. Il s'agit alors 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 1). Celle-là même proposée dans la thèse de Xavier Hinaut [1].

Au centre de la figure 1 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...).

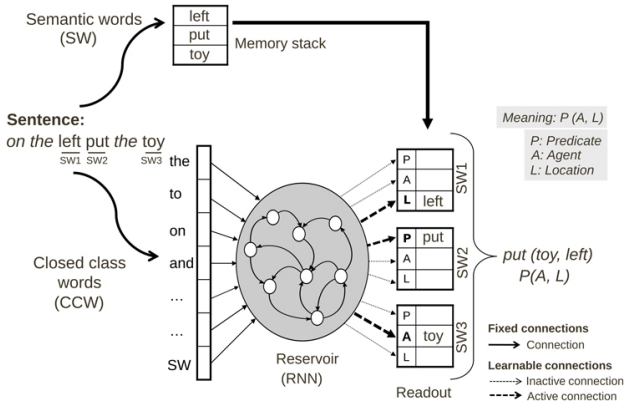

FIGURE 1. Modèlisation du réseau de neurones par Xavier Hinaut

B. *langage à base de boîte*

Afin de permettre à n'importe quel utilisateur de manipuler l'application nous avons pensé qu'utiliser un langage de programmation à base de boîte pourrait être une bonne idée. En effet, 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 gros point fort de ce type de langage est qu'il se rapproche du langage parlé d'une certaine manière.
La majorité de ces langages sont utilisés par des plateformes de programmation afin d'introduire le développement d'applications informatiques à des usagers. Lors de nos recherches nous avons pu en relever trois qui nous ont semblés idéaux pour la réalisation du projet.
Nous allons les lister ci-après :

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 : il s'agit d'une combinaison de Scratch avec son interface graphique et son utilisation simple avec drag-n-drop et Scheme pour sa 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.

PLM : il s'agit d'une 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,...

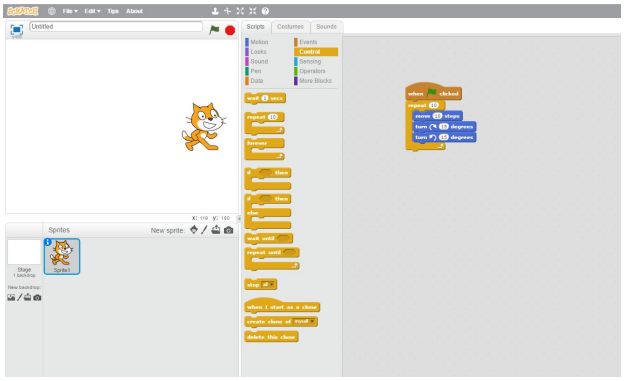

FIGURE 2. Interface de Scratch

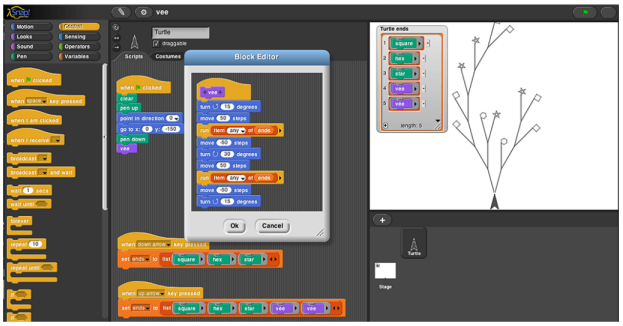

FIGURE 3. Interface de Snap

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 exercices 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, résoudre un problème de type "tour de Hanoï" ou autre.

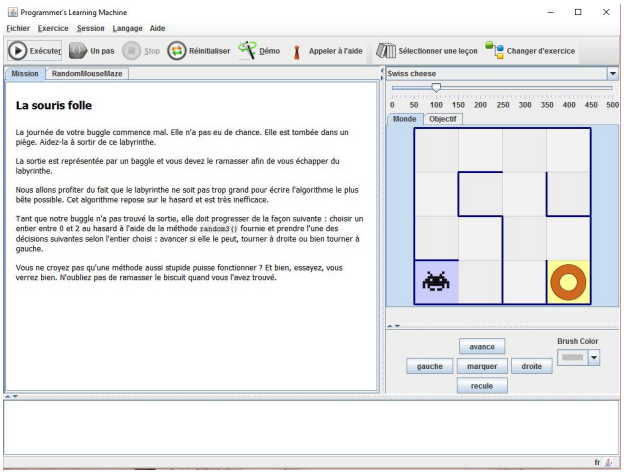

FIGURE 4. Interface de la PLM

C. Réseau de neurones

Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un ensemble d'algorithmes dont la conception de sa structure est très inspirée de celle des neurones du cerveau. Les neurones de ce réseau ont pour objectif d'apprendre au fur et à mesure de l'éxecution de cas de tests. On peut dire tout simplement qu'il s'optimise via des méthodes d'apprentissage. Elles sont généralements probabilistes.
L'idée lors de la réalisation d'un réseau de neurones et surtout lors de son utilisation est donc d'avoir à sa disposition un nombre important de tests pour pouvoir l'éprouver au maximum. Ce n'est que lorsque toutes ces expérimentations seront réalisées que nous pourrons être en grande partie assurés du fonctionnement du réseau de neurones et donc l'utiliser sur un cas plus concret.

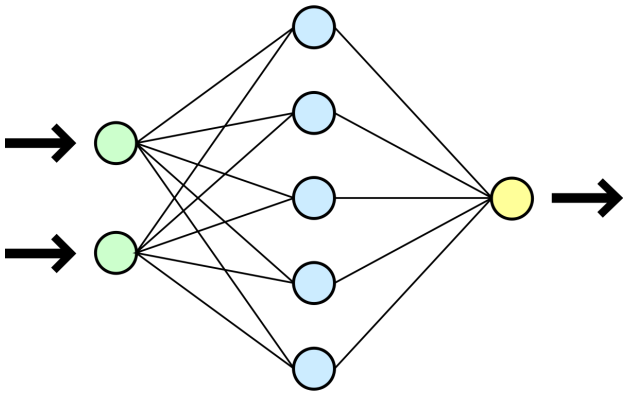

FIGURE 5. Vue simplifiée d'un réseau de neurone artificiel - tirée de wikipédia

III. 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 éprouver 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 celles-ci 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 passé 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. Ensuite, il nous a fallu étudier 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 un certain nombre de fonctionnalités pour répondre à l'ensemble des attentes de l'utilisateur.

IV. MÉTHODOLOGIES

Ici nous allons présenter l'évolution de notre raisonnement tout au long de ce projet. En effet, tout projet de recherche suit un déroulement bien précis : analyse des solutions possibles, mise en place d'une solution choisie et étude des résultats. Dans notre cas notre analyse a porté sur le choix d'un langage de programmation à base de boîtes, élément nécessaire car l'application devait être destinée à des utilisateurs dit "naïfs", c'est-à-dire n'étant pas forcément issue du milieu de la programmation. Or nous souhaitons proposer une interface web offrant la possibilité à ces mêmes utilisateurs de réaliser leur propre vidéos. Ils nous faut donc nous baser sur une plateforme d'apprentissage qui propose à la fois ce type de langage mais aussi une visualisation graphique du résultat pour pouvoir réaliser notre application.
Une fois ce choix fait, il a été nécessaire d'étudier plus en profondeur la plateforme en question pour enfin pouvoir implémenter nos propres fonctionnalités.

A. Etude des solutions possibles

Comme explicité précédemment, le début du projet visait à réaliser un état de l'art des langages de programmation à base de boîtes ainsi que des plateformes qui les utilisent. L'utilisation de ce type de langages avait deux enjeux : le premier était de permettre à un utilisateur lambda, non habitué à l'utilisation de langage de programmation classique tel que le Java ou encore le C, de pouvoir coder ses propres séquences d'instructions. Le second enjeux était de pouvoir disposer d'un langage qui se rapproche le plus possible du langage naturel, car d'une part il s'agit du plus couramment utilisé mais aussi parce qu'il sert finalement de base de recherche à l'équipe Biscuit.

Dès le début de notre étude, notre choix s'est tourné vers quelques uns d'entre eux, que nous connaissions pour en avoir déjà entendu parlés ou qui nous ont été proposé par nos encadrants. Nous nous sommes donc décidés à les étudier en détails afin de déterminer leurs points forts et leurs points faibles pour finalement être en mesure d'en choisir un qui sera utilisé pour le développement de notre application.
Nous en avons retenu majoritairement trois qui aurait pu convenir : Snap, Scratch et la PLM. Snap et Scratch proposaient tous deux des environnements graphiques très développés à base de boîtes et de flèches. La PLM quant à elle restait plus proche de la programmation classique mais offrait la possibilité de rédiger son code via le langage naturel. Tout trois proposaient un environnement de visualisation du résultat de l'exécution.

Pendant toute une phase nous avons éprouvé chacune de ces plateformes afin de déterminer comment elles fonctionnaient en terme de compilation, d'exécution, de vérification de code, etc. Ainsi nous étions à même de pouvoir faire le meilleur choix possible. En effet, nous devions par la suite programmer en usant des technologies de la plateforme en question, il était donc nécessaire, avant de se lancer, d'avoir une connaissance approfondie des technologies employées pour ne pas tomber dans des voies sans issues.
Il a fallut choisir l'une de ses technologies et ce en se basant sur certains critères que nous avions évalués : la facilité de programmation pour les utilisateurs, la facilité de modification des technologies ainsi que la disponibilité des sources et des aides.

Nous avons finalement retenu la PLM. Tout d'abord parce qu'il s'agissait d'une technologie qui était proche de nous (il s'agit d'une plateforme de programmation développée au LORIA par une autre équipe de recherche), ensuite par le fait que le langage de programmation pour les usagers était basé sur le langage naturel donc très facile d'utilisation. Enfin parce que les sources étaient accessibles et assez régulièrement mises à jour.

B. Analyse détaillée de la PLM

Une fois notre choix concernant la plateforme d'apprentissage fais il nous a fallu étudier plus en détails le fonctionnement de la PLM pour déterminer de quelles manières nous allions pouvoir l'altérer le moins possible tout en ajoutant toutes les fonctionnalités demandés.
Nous avons notamment pu observer que l'application web WebPLM était finalement une sorte de surcouche web installée au-dessus de l'application PLM développée en java. Notre analyse a donc dû être réalisée à deux niveaux : d'abord l'étude du code réalisé en Java et qui sert de base au programme, puis l'étude des moyens mis en oeuvre pour permettre à une application locale de devenir une application web.

De manière générale, la PLM et la WebPLM sont organisées selon une structure de code plutôt complexe lorsqu'on l'observe à première vue. Les classes sont organisées d'une certaine façon dans une architecture de fichier particulière. Nous avons donc passé un temps plus que conséquent à étudier le code de la PLM.

Notre objectif dans cette première phase d'étude était surtout de comprendre comment ajouter un nouvel exercice et comment faire que cet exercice fonctionne de la même manière que ceux déjà implémentés dans la PLM.
Finalement, après une réunion avec nos encadrants de PIDR, ceux-ci nous ont proposé d'avoir accès au travail d'étudiants de l'année passée qui avaient travaillé sur un projet similaire et qui, par conséquent pourrait nous aider à avancer dans notre compréhension du code. Nous avons donc pu observer de quelle manière ils avaient altéré la PLM, ainsi ils nous sera possible de faire de même pour notre propre projet. De plus, nous avons eu la possibilité d'étudier les recherches qu'ils avaient mener sur le sujet. Après une étude approfondie de leur travail, nous avons fini par prendre la décision de contacter ces étudiants afin de leur demander plus de détails et éventuellement leur exposer notre sujet et ainsi obtenir leurs avis et conseils. Après une réunion avec eux, nous avons finalement compris comment faire pour modifier la PLM, l'application en Java, afin d'être en mesure de réaliser nos propres exercices. Nous venions de trouver une solution à l'un de nos problèmes principaux.

Après avoir appréhendé le fonctionnement de la PLM, il nous fallait essayer de comprendre un certain nombre d'aspects de la WebPLM puisqu'il s'agissait du deuxième élément qu'il nous fallait savoir manipuler pour pouvoir mettre en oeuvre notre application. Il nous était nécessaire de déterminer un certains nombres de notions : comment celle-ci communiquait avec l'application en locale, comment elle fonctionnait en interne (pour valider le code, pour réaliser l'exécution sur la sortie graphique, etc.). Si nous obtenions une réponse à ces diverses questions, nous serions en mesure de réaliser les fonctionnalités qui nous étaient demandées pour notre application. Nous avons alors pris la décision de contacter Nicolas Mathieu un chercheur du LORIA qui travaillait sur la WebPLM afin de lui apporter des modifications. Cette entrevue nous a permis de comprendre de quelle manière nous pouvions altérer la WebPLM pour notre utilisation personnelle mais il nous a également donné quelques pistes à explorer pour pouvoir aisément développer nos fonctionnalités en se basant sur l'interface web proposée par la WebPLM.

Maintenant nous disposons des connaissances suffisantes sur le code de la PLM et de la WebPLM. Les diverses recherches que nous avons menées ainsi que les discussions que nous avons pu avoir sur le sujet nous ont permis de bien mieux savoir quelle direction il nous fallait emprunter pour mener le projet à bien.La prochaine étape consiste à nous organiser pour déterminer l'ordre dans lequel les fonctionnalités seront à implémenter.

C. Mise en place de nos fonctionnalités

Nous avons pris le parti d'organiser la réalisation de notre application sous forme d'objectifs classés par ordre d'importance et de dépendance. Ainsi nous étions en mesure d'avancer de manière itérative et d'avoir, à la fin de la réalisation d'un objectif, une application fonctionnelle disposant de quelques fonctionnalités. Le but étant qu'à la fin du développement, nous ayons une interface web utilisable et disposant de toutes les fonctions prévues.
Nous nous sommes donc organisés de la manière suivante :

tout d'abord nous avons choisi de travailler la fonctionnalité principale, sur qui reposait l'application, la génération de vidéo à partir du code écrit dans la PLM. Une fois cette partie terminée, nous sommes passés à la réalisation de la page pour permettre d'enregistrer des commentaires. Enfin nous avons ajouté une page d'administration pour offrir la possbilité à l'équipe Biscuit de récupérer facilement tous les commentaires.

Pour conclure, nous pouvons dire que nous avons essayé tout au long de ce projet d'adopter une démarche de recherche. Nous avons débuté en réalisant un état de l'art pour avoir plus de connaissances sur les différentes notions abordés et qui n'étaient pas forcément lié à l'informatique. Par la suite, nous avons fait un certain nombre de recherches dans le but de déterminer quelle base nous devions prendre pour notre application. Ensuite, une fois le choix d'une solution fait, nous avons réalisé une étude approfondie de celle-ci afin d'en connaitre ses limites et de pouvoir au mieux la manipuler pour développer le projet. Enfin nous avons appliqué l'ensemble des connaissances obtenues pour développer une application qui offre toutes les fonctionnalités demandées par nos encadrants.

V. Aspects techniques

Dans cette partie, nous allons mettre en avant les technologies que nous avons du utiliser et maîtriser au cours de ce projet. Tout d'abord nous allons présenter les outils que nous avons dû analyser afin de comprendre leur fonctionnement. Ensuite nous évoquerons les langages et méthodes que nous avons mis en oeuvre pour développer nos propres fonctionnalités.

A. Présentation de l'existant

1) La PLM: Telle que nous l'avons importée, la PLM contenait une dizaine de leçons, telles que "recursion", "buggleWorld" ou encore "sort". Ces leçons contenaient plusieurs exercices permettant d'apprendre comment faire de la récursion, du tri, etc. Chaque exercice contenait un objectif à atteindre afin de finir l'exercice une fois le but atteint.
Le binôme de l'année dernière, que nous avons rencontré avait ajouté une leçon afin de créer des exercices bac-à-sable. Cependant, l'objectif n'était pas retiré et les exercices avaient donc une fin.

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, les murs qui l'entourent. Il est également possible de définir le nombre de Buggles présents sur la carte et leur apparence.

2) la WebPLM: Telle que nous l'avons importée, la webPLM fonctionnait pour chaque leçon de la PLM. Une page d'accueil recueillait la liste de toutes les leçons possibles afin de permettre à 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.
Le binôme de l'année dernière avait implémenté une petite case en bas de chaque exercice (en plus de ceux qu'ils avaient créés) afin de permettre à l'utilisateur de décrire en langage naturel ce qu'il avait fait. Comme par exemple "le buggle a trouvé un biscuit" ou encore "le tableau a été trié".

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 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.

La webPLM est basée sur plusieurs 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. Un acteur n'expose donc pas son état interne au monde extérieur, évitant ainsi la nécessité de se synchroniser lors de l'accès à des données partagées (base de données, leçons de la PLM, etc.). 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, 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. Les fonctionnalités que nous avons implémentées

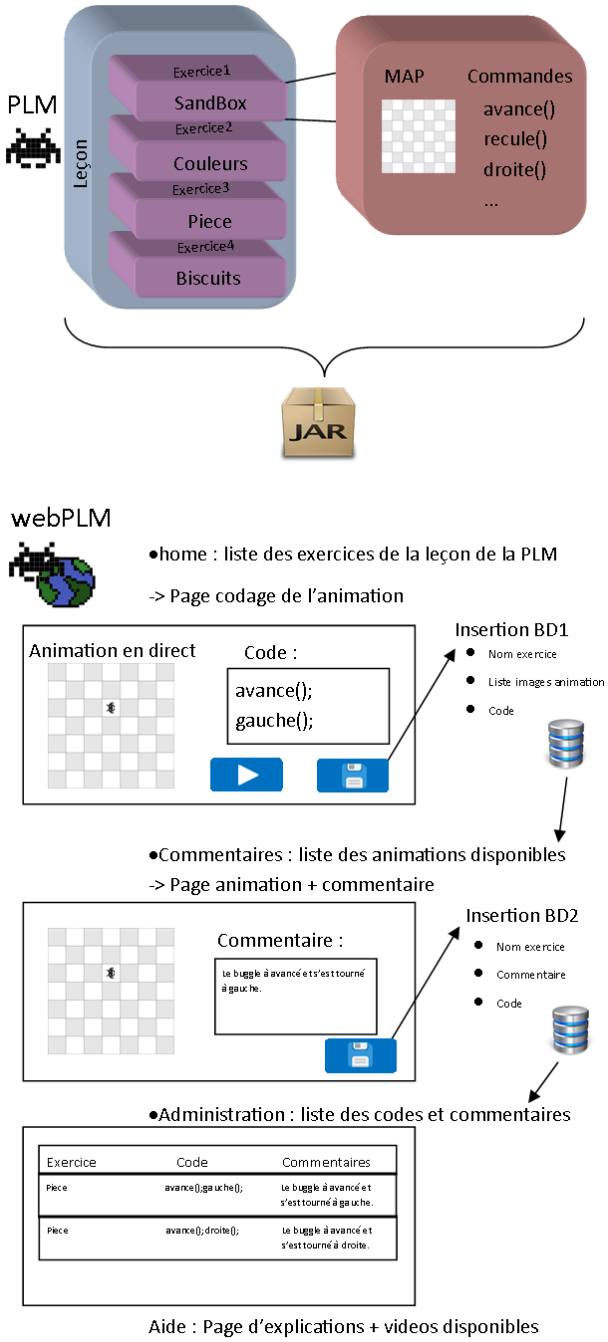

FIGURE 6. Schéma de nos réalisations

Le schéma de la figure 6 présente la structure de notre projet. Tout d'abord, il y a la PLM. Comme présenté précedemment, celle-ci contient énormément de leçons et d'exercices. L'idée était de réaliser des exercices sans objectifs (des sortes d'exercices bac-à-sable) afin de donner le plus de libertés possibles à l'utilisateur. Nous avons donc utlisé les exercices codés par le binôme ayant eu le même sujet de PIDR l'année dernière : un exercice de type SandBox, un ayant des murs qui empêchent la buggle de passer, un ayant des couleurs au sol afin d'exprimer des sentiments, des impressions et enfin un ayant des biscuits au sol.

Nous avons revu la structure de ce code qui ne respectait pas l'architecture globale de la PLM, nous empêchant ainsi d'avoir accès directement aux exercices plutôt qu'à la leçon uniquement.

Le code de la PLM génère donc plusieurs fichiers .jar que nous insérons dans la webPLM. Ainsi, il ne nous restait plus qu'à exploiter les fonctionnalités des exercices pour obtenir une animation codée par un utilisateur.

La suite du schéma expique le fonctionnement de chaque page. Une fois que l'utilisateur est sur la page "home" (la page principale où la liste des exercices est montrée), il peut cliquer sur l'un des exercices pour coder son animation. Sur la page exercice, l'utilisateur peut coder ce qu'il souhaite et visualiser le résultat grâce à un bouton "Exécuter". Il peut ensuite sauvegarder son animation une fois qu'elle lui convient en appuyant sur le bouton "Enregistrer". Ce dernier va, à l'aide du framework akka, envoyer à la base de donnée les informations qui nous intéressent. Soit le nom de l'exercice, la liste des images d'animations et le code qui a généré cette animation. 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.

La page "commentaires" charge une liste de toutes les animations disponibles (elle va donc chercher tous les éléments de la base de donnée précédente). L'utilisateur peut alors cliquer sur l'une des animations afin de la commenter. 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.

La page administrateur permet de visualiser les résultats : l'association d'un code à une description. Pour cela, nous devons afficher tous les éléments de la deuxième BD que nous mettons à jour dans la page précédente.
Enfin, une page d'aide est disponible afin de guider les utilisateurs sur ce qu'ils doivent faire et comment. Des vidéos explicatives ont été ajoutées.

Les 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 (figure 7).

FIGURE 7. Schéma présentant la structure que nous avons utilisé pour accèder aux bases de données

Prenons un exemple pour bien comprendre comment cela est structuré. La page animations.html souhaite afficher la liste de toutes les animations disponibles dans la base de données. Le controller de cette page, animations-controller.js, va envoyer un message en utilisant le framework akka. Ce message ne comprendra pas d'arguments et utilisera la commande "getAllBD1" par exemple. Ce message sera reçu par le scala qui va exécuter la commande. Celle-ci va chercher toutes les informations de la BD1 et va les renvoyer, encore une fois en utilisant le framework akka. Le message comprendra comme argument la liste des entrées de la BD1 et la commande "ListAll(list)" par exemple. Le javascript va alors recevoir ce message et exécuter la commande. Il va ensuite utiliser le fichier html (remplir les bonnes balises) pour afficher les boutons permettant d'accèder à chaque animation.

VI. RÉSULTATS

Dans ce projet nous avons donc mis en place une interface web offrant la possibilité aux utilisateurs, par le biais d'un langage de programmation à base de boîte, de réaliser des courtes séquences vidéos. Ces vidéos peuvent alors être commentés par d'autres utilisateurs et ce dans le but de réaliser un corpus. Celui-ci permettra par la suite à l'équipe Biscuit d'avancer dans leurs recherches sur les mécanismes cognitifs et notamment utilisés le réseau de neurone de Xavier Hinaut [1].

En ce qui concerne les résultats obtenus, notre application réalise les tâches qui lui ont été confié correctement. Elle propose des environnements de travailles de type sandbox et 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 au vu du bon fonctionnement de notre application l'équipe Biscuit va donc pouvoir recueillir le corpus dont elle a besoin pour éprouver son réseau de neurones et ce facilement et rapidemment. De plus, l'application étant utilisable par tout type d'utilisateur, ce corpus ne sera pas biaisé par des connaissances informatiques.

VII. 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 ou encore augmenter la taille du buffer. Une autre voie de développement pourrait être l'ajout de plus de possibilité 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,...

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()” de 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. Auncun commentaire ne pourra être fait sur ces instructions puisqu'elles ne sont pas visibles.

En plus de celles évoqués précedemment, il y a bien évidemment plein d'autres manières de faire évoluer cette application.

Au vu des éléments exposés précédement ils 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.

VIII. 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,... Nous avons également pu mettre en oeuvre un certain nombre de technologies que nous n'aurions peut être jamais utilisées lors de notre formation.

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

BD Base de données. 6

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. 1, 7

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

PLM Programmer's Learning Machine: Outil interactif et graphique pour apprendre à coder. Développé par une équipe du LORIA. 2, 5, 8

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

webPLM Interface web de la PLM. Développé par une équipe du LORIA. 5, 8

ANNEXES

Capture d'écran de l'éditeur de code :

FIGURE 8. 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.

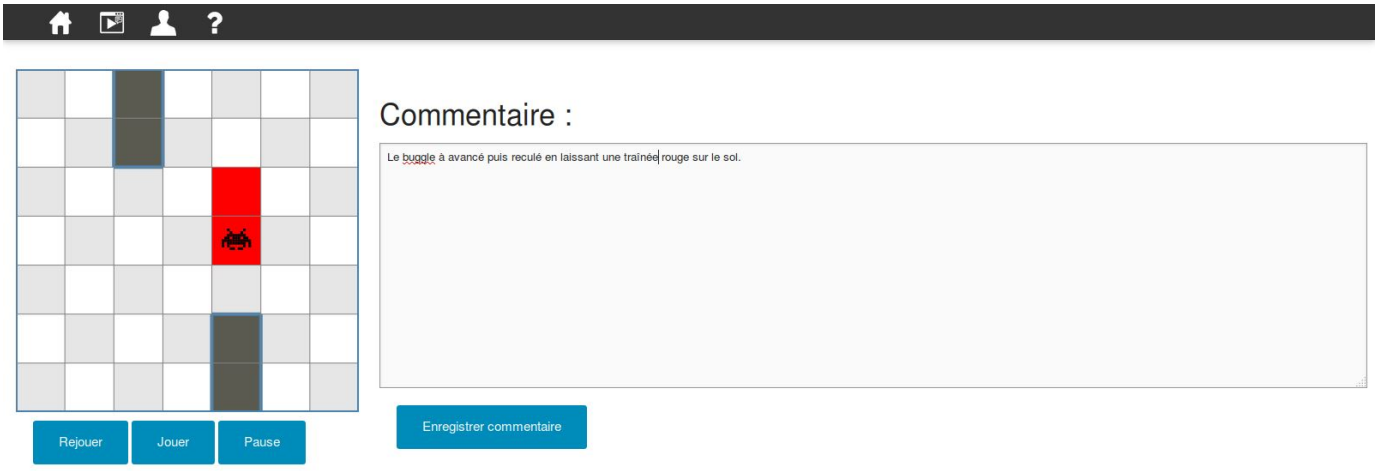

FIGURE 9. Page permettant de commenter une petite animation

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 10. Page permettant de visualiser les commentaires et les codes