SmartInteract

Démarche scientifique

Alexandre Bolot - Laura Lopez - Alexanne Masson - Théos Mariani

État de l'art

SmartInteract est un outil d'aide au développement pour les jeux possédant des dialogues que l'on souhaiterait rendre plus réalistes et vivants en se basant sur du NLP et fonctionnant comme une API. Plus précisément, nous séparons deux aspects distincts au sein du projet :

- Côté Développeur: Le moteur de jeu fournit la possibilité au joueur de dialoguer par saisie clavier avec les personnages enregistrés dans le jeu. Puis il renvoie une réponse adaptée et précise permettant d'avancer dans le jeu.

- Côté Désigner: Une interface graphique qui permet au développeur de créer des personnages et leur attribuer des connaissances. Elle permet aussi de visualiser des logs du déroulement d'interactions en direct, d'utiliser un outils de debug qui permet de dialoguer avec un personnage sans passer par le jeu et de récupérer des statistiques liées aux personnages.

L'innovant

- Utilise des technologies récentes de NLP qu'il met au profit de jeux vidéo ce qui n'a aujourd'hui jamais été réalisé.
- Une solution accessible à tous les langages dans les jeux possédant des dialogues.
- Un projet qui ne nécessite aucune connexion obligatoire.
- Interface de monitoring du moteur de jeu proposée au développeur.

État de l'art

Nous cherchons donc à garantir l'innovation de notre projet en nous renseignant sur :

- L'intégration d'un moteur de jeu avec possibilité de multi langages
- Les technologies récentes de NLP mises au profit de jeux vidéo.
- Les interfaces de monitoring des moteurs de jeu proposées au développeur.

Pour cela, nous avons établi nos recherches sur les diverses plateformes énoncées.

App Store

NLP game
Game monitor
Monitoring

Google Play

NLP game
Game monitor
Monitoring

Mots clés:
NLP in Video Games
NLP interactions
Gamification for natural language processing
Game monitoring tool

Philippe Collet
Pour l'architecture du projet et les technos

Erick Gallésio
Pour le NLP, les technos et la contextualisation de notre projet

Positionnement technique

Au niveau des technologies, nous avons défini que le code de nos composants sera réalisé en Java car c'est le langage avec lequel nous avons le plus d'affinité, qu'il est performant, et que fournissant une interface d'interopérabilité, nous n'avons pas à nous soucier de problèmes de compatibilité. Nous avons également besoin de faire du traitement de langage naturel, et de communiquer de manière distante hors ligne avec les systèmes externes. Pour cela, nous avons effectué les études suivantes :

ORBs	Langage	Spécification	Héritage
CORBA	C, C++, Java	Langage dédié IDL	Héritage d'interfaces
DCOM	C++	Utilise protocole RPC	Composition d'interfaces
Java RMI	Java	Model objet Java	Héritage de classes

Notre choix se porte sur CORBA car :

- Nous avons besoin d'un système opérant inter-langage pour permettre à notre moteur d'être utilisé par des jeux en C, C++ et Java.

- Le langage de description IDL nous permet de facilement générer des Stubs dans les trois langages cités ci-dessus, avec un seul fichier de description.

Positionnement technique

Au niveau du traitement automatique de langage naturel, nous avons choisi Spacy pour ses performances, son panel de langues et la qualité de la documentation en ligne.

Outils	Langage	Précision	Vitesse	Langues reconnues
Spacy	Python/Cython	92,6	13,963	De, El, En, Fr, It, Nl, Pt, Sp
Natural Language ToolKit	Python	n/a	n/a	Beaucoup de langues
Apache Lucene	Java	n/a	n/a	n/a
Stanford Core NLP	Java	89,6	8,602	Ar, De, En, Fr, Sp, Zh
Clear NLP	Java	91,7	10,271	Ar, En, Hi, Ur, Zh
MATE	Java	92,5	550	De, En, Fr, Zh
Turbo NLP	C++	92,4	349	n/a

Vitesse en mots par seconde
Ar : Arabe
Zh : Chinois
Hi : Hindi
Ur : Ourdou
El : Grec

Critères d'évaluation

Intégration

Notre moteur de jeu doit pouvoir être intégré à un jeu réel utilisant la saisie utilisateur. Notre moteur doit pouvoir être accédé depuis plusieurs langages via l'intermédiaire de CORBA.

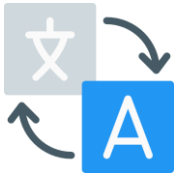
Évaluation du critère : Nous avons prévu d'intégrer notre moteur avec au moins un jeu open-source déjà existant. Nous voulions au départ créer notre propre petit jeu mais cela apportait 2 contraintes : le manque de temps et le fait que ce jeu aurait été orienté dans le but d'être facilement utilisable par le moteur de manière consciente ou inconsciente.

Efficacité

Nous devons garantir l'efficacité du NLP afin que le joueur reçoive une réponse adaptée, précise et rapide.

Évaluation du critère :

Utiliser des benchmarks. Par exemple pour montrer le temps de réponse de notre moteur, qui doit être inférieur à une seconde (le joueur ne doit pas attendre).



Langage

Le moteur doit pouvoir accepter un niveau de langage courant et sans faute d'orthographe.

La tolérance aux fautes et à un vocabulaire familier ou soutenu sera ajouté dans de futures versions.

Évaluation du critère :

L'utilisateur peut saisir au clavier ce qu'il souhaite communiquer, le personnage lui répond une phrase adaptée. L'analyse syntaxique détecte les mots clé au sein de la phrase du joueur.