Document de travail - Workspace, Textspace, Workers

Fernandes-Pinto-Fachada Sarah,
Schrottenloher André,
Angibault Antonin,
Hufschmitt Théophane,
Cao Zhixing,
Boisseau Guillaume

12 novembre 2014

Important

MODIDIFIEZ CE DOCUMENT AUTANT QUE VOUS VOULEZ SELON VOS IDÉES DU MOMENT, C'EST QUELQUE CHOSE DE FULLY COLLABORATIF

1 Présentation

Le Workspace est la structure la plus importante ; plus encore que le réseau de concept. Il nous faut la définir rigoureusement, ainsi que ce que font les workers, afin d'être sûrs à quoi nous arrivons.

C'est aussi ici que nous allons le plus nous démarquer de Copycat notamment, puisque nous voulons faire du résumé automatique : pour ce qui était du réseau de concepts, nous étions encore dans la continuité.

Cahier des charges du workspace

- Contenir des instances des nœuds du réseau de concepts (ie : des concepts signifiants) ;
- Reprendre les informations importantes du texte ;
- Ajouter de l'information ne provenant pas directement du texte, mais jugée importante ? A voir ! (Ex. trivial des nœuds fortement activés dans le RC)
- Ne pas contenir d'information trop évidente : ex. il ne faut pas répéter que Wayne Rooney est un joueur de football, même si c'était un sous-entendu du texte. C'était sous-entendu justement parce que ça ne sert pas de le répéter, a priori ;
- Contenir une information exploitable ;

— Tracer du début à la fin du programme le rôle grammatical des concepts tirés du texte, dans leur contexte. Exemple : Wayne Rooney est un sujet. Il faut savoir tout au long du programme, y compris lorsque le nœud Wayne Rooney sera créé dans le Workspace, que c'est un sujet. Cela permet de ne pas jeter de l'information par la fenêtre.

Cahier des charges des workers Il nous en faut pour :
— Créer des instances de nœuds du RC dans le W ;
— Activer des nœuds du RC ;
— Faire des liens entre les nœuds du W ;
— **Prendre en compte toute l'information grammaticale/syntaxique** et la transférer au W, car elle n'apparaît pas dans le RC. Exemple : un worker spécial qui crée un sujet (Wayne Rooney) dans le W, et va ensuite spécifiquement chercher "ce que fait Wayne Rooney". Un worker spécial qui crée un objet (World Cup) dans le W, et va ensuite chercher "ce qui arrive à" ou "ce qu'on dit sur" World Cup.

2 Idées provenant de Copycat/Bascet

2.1 Copycat

Workspace Le workspace contient des instances du réseau de concepts (Slipnet) organisées en :
— **Description** d'objets
— **Relations** entre objets (lien de succession entre lettres par ex.)
— **Groupes** d'objets
— Correspondances entre objets de chaînes différentes ("est analogue à")
— Règles qui décrit le changement entre la chaîne initiale et la chaîne modifiée
— Règle traduite qui décrit comment modifier la chaîne cible (peut être issue de la règle précédente en remplaçant des concepts, par exemple)
Je retiens : description, relations, groupes d'objets.

coderack Le coderack contient les codelets, agents choisis stochastiquement mais pas n'importe comment.
Ces agents (codelets) représentent des pressions.
— Codelets ascendants : pressions présentes dans toutes les situations (trouver des relations...)
— Codelets descendants : pressions provenant de la situation courante
À chaque pas d'exécution, un codelet est exécuté et supprimé du coderack. Chaque codelet a une certaine urgence, mais elle ne détermine pas sa priorité (le choix se fait en partie au hasard) (elle représente l'intérêt de sa tâche).
— Les codelets activent les noeuds du Slipnet
— Les noeuds actifs ajoutent des codelets dans le Coderack (descendants, par exemple)
— Les codelets lancent d'autres codelets
— Des codelets ascendants sont créés en permanence
Je laisse pour plus tard le problème de la température.

2.2 Bascet

Agents
- Chacun a son **urgence**
- Chacun a un **nœud père** qui l'a lancé (RC)
- Ils peuvent lire et créer les objets du Blackboard (active les nœuds du RC dont l'objet est une instance)
- **Opinionnent** sur la validité d'un objet

Blackboard Contient des objets, instances des noeuds du RC :
- Contenu
- Père (noeud dont il est instance)
- Importance (dépend du père et des liens avec les autres objets)
- Satisfaction (dépend des agents)
- Eminence (objet à traiter)

3 Notre architecture

Je pense reprendre dans un premier temps un mélange de ces caractéristiques, et adapter ensuite à notre problème.

Définition 1 (Textspace) *Le textspace [nom à changer, par pitié !] est un espace intermédiaire dans lequel l'information est stockée sous forme d'arbres sémantiques. Cette information est :*
- *Décomposée par les workers ;*
- *Envoyée par les workers dans le workspace, avant d'être reconstruite.*

Définition 2 (Workspace) *Le Workspace est une structure constituée de :*
- *Nœuds issus de l'instantiation de concepts ;*
- *Relations entre ces nœuds.*

Définition 3 (Nœud du Workspace) *Un tel nœud comporte :*
- *Un **nœud père** dans le RC et une étiquette identique à son père ;*
- *Une **satisfaction** calculée par les workers, selon le rôle, et les liens qu'ils ont réussi à créer (par exemple, pour un sujet, on s'attend à ce que celui-ci agisse) ;*

Moins le nœud est satisfait, plus il faut chercher, donc lancer des workers qui traitent de lui OU augmenter l'urgence de ces workers.

Définition 4 (Lien du Workspace) *Lien entre deux nœuds du workspace. Ce lien exprime :*
- *Une relation (sujet-objet, objet-caractéristique)*
- *Un concept (le lien est donc labellisé par un nœud du RC)*

Définition 5 (Worker) *Un worker correspond à une tâche. De manière générale, il agit à la fois sur le Workspace et sur le RC. Un worker comporte toujours :*

— *Une urgence ;*
— *Une mission définie ;*
Il existe différents types de workers.

Définition 6 (Worker descendant) *Le texte initial (phrase) étant entré dans le textspace sous forme d'arbre sémantique, un worker descendant est un worker qui se charge de décomposer cette structure pour en déduire des relations simples entre objets.*
Les workers descendants doivent dépendre foncièrement de la représentation de la phrase.

Définition 7 (Worker ascendant ou Builder) *Un builder, comme son nom l'indique, est un worker qui travaille sur le workspace pour construire des structures. Il peut :*
— *Créer un nœud dans le workspace ;*
— *Créer un lien dans le workspace ;*
— *Rechercher des liens possibles ;*
Le builder s'appuie sur différentes informations : l'activation du RC et le texte initial (décomposé grâce aux workers descendants).

4 Liste actualisée des workers et détail de leur activité

Remarques générales :
— Moins un nœud est satisfait, plus il doit générer de workers constructeurs
— La satisfaction d'un nœud doit augmenter lorsque quelque chose est construit sur ce nœud (par exemple lorsqu'on crée des liens)
— Les workers les plus importants sont ceux qui activent les concepts
— Ensuite viennent ceux qui construisent quelque chose
— Ceux qui font des calculs généraux
— etc

4.0.1 Worker

Classe-mère. Augmente le temps lors du lancement du worker, et affiche un message si cette fonction est activée.

4.1 Workers qui n'agissent directement que sur le RC

4.1.1 Activate

Urgence : 100 Active un nœud du RC avec un certain montant (l'activation d'un nœud ne peut pas dépasser 100).
— Si le nœud activé dépasse un certain seuil (80), push un WriteNode correspondant ;
— Dans tous les cas, push des workers Compute pour tous les nœuds dans les liens de sortie.

4.1.2 Compute

Urgence : 50
Recalcule l'activation d'un nœud du RC. Push des workers Compute pour tous les nœuds dans les liens de sortie.

4.2 Workers descendants, qui agissent sur le textspace

Ce sont eux qui vont chercher les concepts du RC à partir des mots de la phrase, avant d'envoyer des informations intelligibles dans le workspace. Ce sont eux qui initialisent la lecture de la phrase, envoient des workers activate pour travailler sur le RC.

4.2.1 ReadPhrase

Urgence : 100
Lit une unité complète.
— Pour chaque mot, activer le RC
Il n'est pas exclu de lire plusieurs phrases en même temps ! !

4.2.2

4.3 Builders : workers ascendants (up)

Il y a trois possibilités :
— Soit on active le RC quand on construit des structures ;
— Soit, au contraire, on le désactive ;
— Soit on n'active rien du tout.
Je préfère la première possibilité : cela donne plus de chances d'agrandir les structures qu'on est en train de construire. En revanche, il faut activr avec modération, sinon on va commencer à faire n'importe quoi.

4.3.1 NodeBuilder

Worker chargé de créer un nœud.
— On lui donne un nœud du RC très activé
— Si ce nœud est déjà instancié, il ne fait rien (à voir : il créé une nouvelle instance ? ? ?)
— Si ce nœud n'est pas instancié, il créé une instance et désactive le nœud
— Il envoie un LinkBuilder

4.3.2 NodeFinder

Worker chargé de trouver un nœud du workspace correspondant à un nœud du RC.
Lorsque nous ferons des instanciations multiples de nœuds du RC, nous aurons bien besoin de lui pour décider à quel nœud du W s'intéresser... Pour le moment, on s'en fiche pas mal.

4.3.3 LinkBuilder

Worker chargé de trouver des liens partant d'un nœud déjà instancié dans le workspace.

Le worker recherche dans le textspace (mettons par exemple qu'on a décomposé la phrase en petits bouts) des liens possibles. Différentes possibilités s'offrent alors :

- S'il trouve un lien avec un nœud instancié et activé dans le RC, il faut :
  - Activer encore ce nœud
  - Créér le lien dans le W
  - Supprimer le lien dans les informations accessibles (à voir)
- S'il trouve un lien avec un nœud instancié mais pas activé dans le RC, il faut :
  - Instancier à nouveau le concept du RC considéré
  - Créér un lien avec cette instance
  - Activer le nœud du RC

  Il faut aussi comprendre que désormais, lorsqu'il sera question de ce concept, c'est à cette instance qu'on fera appel (concrètement, c'est difficile).
- S'il trouve un lien avec un nœud ni instancié, ni activé, il faut :
  - Instancier ce nœud
  - Activer le concept

Dans tous lescas où un lien est créé, la satisfaction des membres du lien augmente.

5 Fonctionnement général

1. La phrase est lue, parsée etc ;
2. L'arbre sémantique correspondant est envoyé dans le workspace ;
3. Un worker initial est appellé sur l'arbre sémantique ;
4. Le worker décortique l'arbre sémantique en activant les uns après les autres les nœuds correspondants du RC ;
5. Les activations se propagent au fur et à mesure et dès qu'un nœud est assez activé, il est instancié, l'ordre de lecture importe donc ;
6. Lorsqu'un nœud est instancié, il est désactivé ;
7. Lorsqu'une relation du RC entre deux nœuds semble pertinente (ie : ces deux nœuds sont activés, ils sont déjà instanciés), on crée un lien dans le workspace entre ces deux instances. Ce peut être une relation conceptuelle (ie : il y a une étiquette sur le lien) mais pas sujet-objet. Il faudra ensuite travailler ça ;
8. Etant donné un nœud qui vient d'être instancié, on crée un worker chargé de trouver des liens partant de ce nœud.
9. Un calcul global de satisfaction des différents nœuds permet de se rendre compte de l'avancée du travail. (Si la satisfaction stagne, on n'a plus qu'à arrêter)