Rapport de stage - Théorie des jeux et évaluation des dialogues pour succés

Sablayrolles Louis

7 avril 2017

Table des matières

1 Base de travail

1.1 Etat de l'art

La base de ce travail est fondé sur l'article [VOM12]. Il nous explique la méthode Overt Display of power (ODP) dans le contexte d'un système d'apprentissage supervisé. Cette méthode consiste à appliquer des poids sur chaque échanges afin de calculer une sorte de récompense que chaque participant va gagner. Celle-ci est calculée en utilisant le contexte de l'échange et non le type de phrase. Dans cet article un système permettant d'automatisé les tags ODP est présenté ; il associe à chaque phrasede l'échange un poids basé sur un apprentissage supervisé utilisant des lemmes. Pour résumer, ce système calcule les réponses possibles pour chaque phrase et les comparent à la vrai réponse afin de désigner au final un gagnant.

2 Modélisation

2.1 Dépendances au sein d'un débat

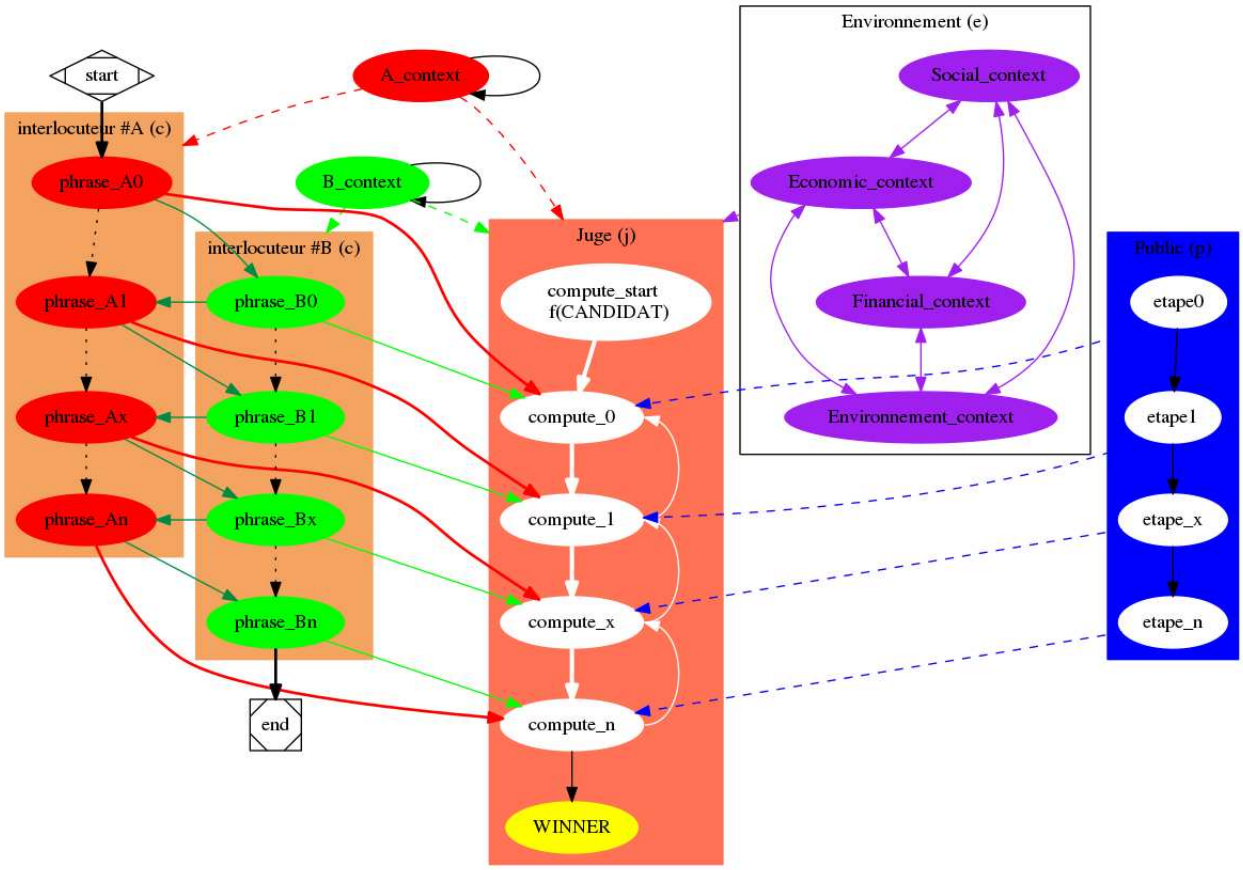

FIGURE 1 – Représentation d'un débat public entre deux personnes avec un juge impartial

La figure ci-dessus représente un débat public entre deux personnes avec un juge impartial. Les différentes parties présentes sont les deux interlocuteurs #A #B qui représente les deux personnes « s'affrontant » (type c : candidat), le juge (type j) qui évalue la situation de chaque candidats en fonction de leurs propos et des contextes extérieurs tel que l'environnement (e),

le public (type p) qui réagit après chaque échange entre les deux candidats. Le débat en lui même est représenté par les flèches vertes foncées sur le graphe. A la fin du débat le juge qui a attribué des points au deux candidats les somment et donne un gagnant.

2.2 Modélisation mathématique

Types

num : numéro de l'échange

int : evaluation $\begin{cases} > 0 & \text{si evaluation positive} \\ = 0 & \text{si evaluation neutre} \\ < 0 & \text{si evaluation negative} \end{cases}$

string list : phrases énoncés par le candidat
char : lettre identifiant le candidat
int*int : couple de scores obtenu par les deux candidat
bool : indique si un candidat n'a plus rien a dire
react : reaction du public (idem int)

Domaines

$num \in \mathbb{R}^+$
$int \in \mathbb{R}$
$stringlist \in [A - Za - z0 - 9.!?, ; \$]$
$char \in A, B$
$bool \in true, false$

Fonctions generales et typages

compute($num, phrases_X[], phrases_Y[], letterFirstCandidat, result\_oldCompute[], A\_context, B\_context, Env\_context, public_react$);
Description : Calcul a un instant num les scores des candidats.
compute : num -> string list -> string list -> char -> num*int*int list -> context -> context -> context -> react -> num*int*int

winner($last\_result\_compute, Env\_context, public\_react$);
Description : Compare les scores finaux obtenus pour désigner un vainqueur.
winner : num*int*int -> char

Fonctions auxiliaire d'évaluation

eval($contexte$);
Description : Evalue un contexte en en donne une note.
eval : contexte -> int

moyenne($contexteCandidat, contexteAdversaire, Env\_context$);
Description : Fait les moyennes de tous les contextes et en produit un contexte global.

moyenne : contexte -> contexte -> contexte -> int

nuance($contexte, public\_react$);
Description : Nuance un contexte suivant la reaction d'un public.
nuance : contexte -> react -> int

pondere($num, phares\_X[], eval$);
Description : Pondere les phrases dite par le candidat en fonction du contexte dans lequel elle sont dites.
pondere : num -> string list -> int -> string*int list

interprete($num, phares\_X[], personalContexte, Env\_context$);
Description : Interprete les phrases d'un candidat dans un contexte et un environnement donné.
interprete : num -> string list -> contexte -> contexte -> interpret

getAtmosphere($num, personalContexte, Env\_contexte, Adversaire\_contexte$);
Description : Retourne l'atmosphere generale.
getAtmosphere : num -> contexte -> contexte -> contexte -> interpret

fini($num, lettre$);
Description : Détermine si un candidat a fini et laisse tomber le débat ou si le temps est écoulé (nombre d'échange max).
fini : num -> char -> bool

compare($num, interpretX[], interpretY[]$);
Description : Compare deux interpretation et donne une note en fonction aux deux candidats.
compare : num -> interpret -> interpret -> int*int

mix($result, result\_oldCompute[]$);
Description : Calcul le résultat final en fonction du resultat de la comparaison et des precedants résultats.
mix : int*int -> int*int list -> int*int

add($result, result\_oldCompute[]$);
Description : Ajout le résultat dans les vieux résultats.
add : int*int -> int*int list -> int*int list

Axiomes

Notons par convention :

- c_C : contexte du candidat
- c_A : contexte de l'adversaire
- c_E : contexte de l'environnement
- r : réaction du public
- m : longeur du discours

— n : numéro actuel de l'échange
Ainsi que les ensemble suivant :

— $\mathscr{C}$: l'ensemble des contextes existants
— $\mathscr{P}$: l'ensemble des phrases possibles

$$\forall c \in \mathscr{C}, \text{eval}(c) \in \mathbb{Z}$$

$$tq \begin{cases} \text{eval}(c) > 0 & \text{si c'est bon pour le candidat} \\ \text{eval}(c) = 0 & \text{si cela n'a pas d'influence sur le candidat} \\ \text{eval}(c) < 0 & sinon \end{cases} \tag{1}$$

$$\forall c_C \in \mathscr{C}, \forall c_A \in \mathscr{C}, \forall c_E \in \mathscr{C},$$
$$\text{moyenne} = \frac{\text{eval}(c_C) + \text{eval}(c_A) + \text{eval}(c_E)}{3} \tag{2}$$

$$\forall c \in \mathscr{C}, \forall r \in \mathbb{Z},$$
$$\text{nuance}(c, r) = eval(c) \times r \tag{3}$$

$$\forall n \in \mathbb{N}, \forall m \in \mathbb{N}^*, \forall p[] \in \mathscr{P}^m, \forall c \in \mathscr{C},$$
$$\text{pondere}(n, p[], c) \Rightarrow (\forall i \in \{0, \ldots, m\}, p[i] \rightarrow pond[i] = p[i], c) \tag{4}$$

$$\forall n \in \mathbb{N}, \forall m \in \mathbb{N}^*, \forall p[] \in \mathscr{P}^m, \forall c_C, c_E \in \mathscr{C}^2,$$
$$\text{interprete}(n, p[], c_C, c_E) \in \mathbb{Z} \tag{5}$$

$$\forall n \in , \forall c_C \in \mathscr{C}, \forall c_E \in \mathscr{C}, \forall c_A \in \mathscr{C},$$
$$\text{getAtmosphere}(n, c_C, c_E, c_A) \in \mathbb{Z} \tag{6}$$

$$\forall l \in \{"a", "b"\}, \exists n \in \mathbb{N}^*$$
$$\text{fini}(n, l) = true \tag{7}$$

$$\forall n \in \mathbb{N}, \forall intX \in \mathbb{Z}, \forall intY \in \mathbb{Z},$$
$$\begin{cases} \exists a, b \in \mathbb{Z}, a > b \wedge \text{compare}(n, intX, intY) = (a, b) & intX > intY \\ \exists a, b \in \mathbb{Z}, a = b \wedge \text{compare}(n, intX, intY) = (a, b) & intX = intY \\ \exists a, b \in \mathbb{Z}, a < b \wedge \text{compare}(n, intX, intY) = (a, b) & \text{sinon} \end{cases} \tag{8}$$

$$\forall a, b \in \mathbb{Z}, \forall n \in \mathbb{N}, \forall l \in \mathbb{Z}^{2^n}, \exists c, d \in \mathbb{Z},$$
$$\text{mix}((a, b), l) = (c, d) \tag{9}$$

$$\forall a, b \in \mathbb{Z}, \forall n \in \mathbb{N}, \forall l \in \mathbb{Z}^{2^n},$$
$$\text{add}((a, b), l) \Rightarrow l[n] = (a, b) \tag{10}$$

Bibliographie

[VOM12] Prabhakaran Vinodkumar, Rambow Owen, and Diab Mona. Predicting overt display of power in written dialogs. *In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, page 518 to 522, 2012.