Le framework LiteTalk

François Bouchet
LIP6 / SU
francois.bouchet@lip6.fr

26 octobre 2018

SORBONNE UNIVERSITÉ

http://bit.ly/EVIJV-2018-5

LIP6

Framework LiteTalk

- Projet initié par Jean-Paul Sansonnet (LIMSI-CNRS) en 2010
- Libre d'accès pour l'enseignement et la recherche (pas d'exploitation commerciale)

- Toolkit d'interactions textuelles avec des agents conversationnels intégrés à des pages Web
- Moteur de dialogue pour les agents DivaLite

LiteTalk : principes

- Objectif général : conversations sur un sujet ciblé
  (≠ ALICE qui vise à traiter n'importe quel sujet)

- Interactions :
  - Modalités :
    - langue naturelle
    - contrôle/commande (proportion croissante au cours du temps)
  - Objectifs :
    - obtention d'informations
    - demande d'exécution d'actions sur la page

- Connaissances : organisées en topics (sujets de conversation) :
  - l'agent lui-même,
  - différents éléments de la page web,
  - des sujets abstraits…

LiteTalk : structure des interactions

- Interactions textuelles reconnues :
  - Langue : anglais
  - Non sensible à la casse (usa = USA)
  - Ne reconnait que les caractères alphanumériques
- Syntaxe de base :
  - Métaphore de la commande avec N paramètres :
    COMMAND PARAM PARAM …
    ask agent age
    bye

  Commande au début, paramètres non ordonnés
  - Utilisation de la langue naturelle pour reconnaître des variantes :

  « Could you tell me what your age is? »
  = « What is your age? »
  = **ask** agent age
  = **a** agent age (version abrégée)

Types de paramètres

Code	Description
B	nom du chatBot
T	Topic
A	Attribut
V	Valeur (d'un attribut)
F	nom d'une Fonction
O	nom d'une Opération
J	Jugement
R	Référence
_	(pas de paramètre)

LiteTalk : liste de commandes liées aux connaissances

Code	Commande	Paramètres	Description	Réaction du chatbot
a	Ask	T A	Demande de la valeur de T.A	Valeur de T.A
v	Verify	T A V	Demande si T.A = V	Oui/Non
t	Tell	T A V	Informe que T.A = V	Met à jour T.A à V
r	Reply	V | R	Répond à une question avec une valeur / référence	Remerciement
k	Know	R	Demande si connait une référence	Oui/Non
w	Why	_ | T A V	Demande explication de la réaction précédente / pourquoi T.A = V	Description de la raison
p	Possible	F	Demande si une fonction peut être exécutée (disponible, autorisé…)	Oui/Non
h	How	O	Demande quelle fonction peut permettre de réaliser une opération	Description d'une procédure
e	Effect	F	Demande de l'effet d'une fonction	Description d'un effet

e.g. « what is your name? », « can you start the counter? », « how to stop the counter? »

LiteTalk : liste de commandes liées aux opinions

Code	Commande	Paramètres	Description	Réaction du chatbot
j	Judge	R J	L'utilisateur juge qu'une référence est J	Mise à jour du topic de user avec R J
f	Feel	J	L'utilisateur se sent J	Mise à jour de userTopic.feelings avec J
l	Like	R	L'utilisateur aime la référence	Mise à jour de userTopic.likes avec R
d	Dislike	R	L'utilisateur n'aime pas la référence	Mise à jour de userTopic.likes avec R
b	Bravo	_ | R	L'utilisateur félicite le bot (au sujet d'une référence)	Augmente botTopic.mood et merci
c	Criticize	_ | R	L'utilisateur critique le bot (au sujet d'une référence)	Diminue botTopic.mood et désolé
s	Suggest	T F | T F (x_i)	Recommande d'exécuter une fonction d'un topic (avec des arguments)	Merci pour info et exécute F (ou pas)
i	Intent	T F	L'utilisateur informe de son intention d'exécuter F bientôt	Mise à jour de userTopic.intent avec T.F

LiteTalk : liste de commandes liées aux actions et au dialogue

Code	Commande	Paramètres	Description	Réaction du chatbot
y	Yes	_	L'utilisateur répond oui à une question précédente	Mise à jour et prise en compte
n	No	_	L'utilisateur répond non à une question précédente	Mise à jour et prise en compte
g	Greet	B	L'utilisateur salue le bot B (pas forcément but en cours)	Salue en retour (et changement pour le bot B)
q	Quit	_	Quitte l'application	Dit au revoir et ferme l'application
x	eXecute	T F | T F (x_i)	Demande d'exécuter une fonction (avec arguments ou non)	Effectue l'action et informe que c'est fait
m	More	_	Demande de refaire l'action précédente	Le fait
u	Undo	_	Demande d'annuler l'action précédente	Le fait

e.g. « do stop the counter » = « stop the counter », « bye », « hello Fifi »…

LiteTalk : création d'une application web dialogique

1. Chaque application Web utilisant LiteTalk est une page .html devant inclure au moins 2 fichiers JavaScript :
   - `litetalk.code.js` : le moteur de LiteTalk
   - `monAppli.code.js` : les informations propres à l'application

```
<script type="text/javascript" src="../../litetalk.code.js"> </script>
<script type="text/javascript" src="developer.code.js"> </script>
```

2. Définir pour chaque bot myBot (tel que `[["KEY","_htmlprefix"],["VAL", "myBot"]]`) :
   - Une zone de texte nommée Bballoon

```
<textarea id="myBotballoon" cols="30" rows="2" spellcheck="false"
style="overflow:hidden; border:hidden; visibility:hidden; font-
family:Segoe Print;">.</textarea>
```

   - Une image (ou un personnage) nommée Bpict

```
<img id="myBotpict" src="picts/myBot.png" width="120"
height="146" alt="myBot"
onClick="BOT_standardSwitchToChar('myBotBot');" />
```

LiteTalk : page web dialogique (suite)

3. Ajouter une chatbox :

```
<div style="position:relative; left:100px; top:50px;
height:520px; width:500px; border:0px solid red;">
    <input id="litetalkchatbox" type="text" size="55"
value="ask name"
onKeyPress="BOT_chatboxOnKeypress(this,event);">  
     <input type="submit" name="sendButton" id="sendButton"
value="SEND" onClick="BOT_chatboxOnSend('litetalkchatbox');">
 
     <input type="button" value="+"
onClick="BOT_switchTraceBox(this)"><BR>

    <div id="litetalkstatustext" align="left" style= "font-
family:Verdana, Geneva, sans-serif;  font-size:10px;
     width:450px; height:80px; border:0px solid pink;">.</div>
<BR>

     <p> <textarea id="litetalktracebox" cols="55" rows="20"
spellcheck="false" style="visibility:hidden;">.</textarea> </p>
</div>
```

Zone de texte

Bouton d'envoi

+ : affiche debug

Statut
(bots, topics)

Zone de debug

4. Lancer l'application :

```
<script type="text/javascript">
    BOT_theInterface          = "STANDARD";
    BOT_traceFlag             = false;
    BOT_statusFlag            = true;
    BOT_standardFrameBot('myBotBot','visible','4px solid yellow');
</script>
```

Fichier .js propre à l'application

- Dans `monAppli.code.js`, on définit l'ensemble de topics, qui sont les sujets dont peut parler un chatbot

- Un topic est de la forme :

```
var topicName = [array of ELEMENTS];
ELEMENT = [["KEY", "keyname"], [TAG1, expr1], [TAG2, expr2], …, [TAGN, exprN]]
```

  - KEY et sa valeur sont obligatoires
  - Les autres TAGs sont optionnels
  - Il ne peut y avoir qu'un TAG avec un nom donné dans un élément

- Puis on initialise les bots et on déclare les topics :

```
var myBot = new BOT_makeBot("myBot","myBotTopic");
BOT_declareTopics(["userTopic","otherTopic1",…]);

BOT_theBotId = "myBotBot";              // sets current bot id
BOT_theTopicId       = "myBotTopic";    // sets current topic id
BOT_theUserTopicId   = "userTopic";     // sets topic of current user id
```

LiteTalk : tags principaux des topics

- `VAL` : valeur d'un attribut, nom interne d'une fonction…
- `CAT` : catégorie de l'élément :
  - `INFO` = info statique,
  - `VAR` = variable dynamique,
  - `ACT` = action statique,
  - `REL` = pointeur vers un autre topic
- `TYPE` : type de la valeur de l'élément :
  - `STR` = string, `INT` = integer, `BOOL` = booleen,
  - `EXPR` = code à évaluer – par exemple si VAL = « alert(1+1) »
- `ONASK` : réponse à envoyer quand Ask de l'utilisateur (au lieu de juste `VAL`)
- `WHY` : raison pour laquelle `VAL` vaut la valeur actuelle

Ex : `[["KEY", "age"], ["VAL", 20],["TYPE","INT"], ["ONASK","I am twenty year old"], ["WHY","I was born twenty years ago"]`

- `HOW` : explique comment une opération sur l'élément peut être faite
- `EFFECT` : explique l'effet de la fonction de l'élément
- `REVERSE` : fonction inverse de la fonction de l'élément (pour « undo »)

Ex : `[["KEY", "faster"], ["VAL", "func_faster"], ["CAT","ACT"], ["REVERSE","slower"], ["HOW","c"], ["EFFECT","increases the speed of _TN_"]]`

Accès au framework

Téléchargement :

- Framework LiteTalk 1.1

http://fbouchet.vorty.net/classes/evijv/2017/litetalk1.1.zip

Plus d'informations :

- LiteTalk :

http://www-poleia.lip6.fr/~bouchet/diva/litetalkwebsite/litetalk.site.main.html

Mini-projet

1. **Proposer un jeu** nécessitant une interaction dialogique avec au moins un agent :
   - combien d'agents ?
   - quel est le contexte d'interaction ?
   - quel est le but du jeu/de la conversation ?
   - à quoi ressemblera l'interface ?

   Proposition (1-2 pages) à soumettre par mail avant le 01/11/2018
2. **Recueillir un corpus** (au moins quelques centaines de phrases) de phrases attendues dans le contexte de votre application :
   - Par chaque membre du binôme
   - Par une personne extérieure : autre binôme, famille, amis…
3. **Analyser le corpus** : produire grâce à NLTK quelques statistiques sur votre corpus : diversité lexicale, mots les plus courants, etc.
4. **Créer une page web** intégrant un/des objets/agents avec des capacités dialogiques en utilisant **LiteTalk**
5. **Tester** les phrases du corpus

Soutenances individuelles

Séance du 18/01 : 13h30-16h00

- 10 minutes par binôme (créneaux de 15 minutes, ordre à définir)
- 20% de la note de l'UE

- Présentation sur machine :
  - Présentation de **l'application** réalisée
  - Présentation du **corpus** collecté + **stats avec NLTK**
  - **Démonstration** de l'application
  - **Discussion** sur l'implémentation
  - **Retour sur expérience** : difficultés rencontrées, objectifs accomplis/échoués…

- Fournir le code permettant de faire tourner l'application
  - .html (au moins 1)
  - .js (au moins 1 pour l'appli, éventuellement `litetalk.code.js` si modifié)
  - Images
  - Corpus de phrases
  - Rapport d'analyse de corpus (quelques pages)

Exercices

1. Télécharger le framework LiteTalk

2. En reprenant le code du cours et en s'inspirant de l'exemple dans demo_fifi
   - créer une page avec une image représentant un simple agent
   - créer quelques clés avec des tags de différents types (VAL, ONASK, HOW…)
   - Tester que tout fonctionne comme attendu

3. Etendre le vocabulaire reconnu en éditant le fichier litetalk.code.js pour ajouter des synonymes à certaines actions (e.g. « hate » pour « dislike »)

4. Ajouter un 2e chatbot (statique) à la page et gérer le passage de l'un à l'autre (cf. exemple de demo_fifi)

Quelques remarques :

- Attention aux « s » à la fin des noms propres :
  - « Nicolas » est « lemmatisé » en « Nicola »
- Quand on renomme un bot, penser à changer le nom *partout*, y compris dans les images associées
- On peut traduire la base de vocabulaire facilement pour passer au français, mais attention aux accents et tirets (trait d'union)