# Human2Chatbot : Génération automatique de règles pour agent conversationnel ChatScript

Mica Ménard
Université de Nantes
Sous l'encadrement de Nicolas Hernandez

Mai 2018

**Résumé**

Les agents conversationnels à base de règles les plus développés donnent l'illusion d'interagir avec un humain. La performance de ces programmes dépend cependant de la qualité de leurs règles et la conception de celles-ci peut être laborieuse. Dans cette optique, ce travail consiste à proposer un modèle permettant de créer des agents conversationnels de manière automatique en automatisant la génération de leurs règles.

## Table des matières

# Introduction

La motivation de ce Travail d'Étude et de Recherche (TER) est de (1) proposer une approche permettant de générer des règles pour agent conversationnel de manière automatique et reproductible en s'appuyant sur l'observation de conversations entre humains ainsi que de (2) déterminer les problématiques liées à cet objectif.

Pour procéder, ChatScript (Wilcox and Wilcox 2017) est utilisé comme architecture de génération automatique d'agents conversationnels et le corpus Ubuntu (Liu et al. 2016) est exploité en tant que base de conversations entre humains. Les objectifs de cette étude sont détaillés après une revue de l'état de l'art ainsi qu'une présentation des outils utilisés.

## État de l'art

Les agents conversationnels (AC) sont des programmes interagissant en langage naturel avec un ou plusieurs utilisateurs. L'état de l'art comprend différentes approches dans la conception d'AC. Cette section constitue une tentative de synthèse de ces approches.

### Initiative mixte

La demande de l'industrie exige une stratégie de dialogue sous forme d'*initative mixte*, c'est-à-dire un système qui "*permet à l'utilisateur de prendre le contrôle en posant des questions ou en fournissant plus d'informations que nécessaire tandis que le système est capable d'effectuer des propositions et d'assister l'utilisateur dans l'accomplissement ses objectifs*" (Berg 2014).

Afin de remplir cette demande d'initiative mixte, ChatScript (Wilcox and Wilcox 2017) a recours à un système de règles qui détermine une correspondance à l'entrée de l'utilisateur afin de retourner une réponse appropriée.

Une autre stratégie d'initiative mixte consiste à procéder à une analyse linguistique puis à accéder à une base de connaissances (Lester, Branting, and Mott 2004). De même, NADIA (Berg 2015) est un AC qui répond au critère d'initiative mixte en procédant à une analyse pragmatique des actes du dialogue dans l'entrée de l'utilisateur ainsi qu'à une génération de langage naturel pour la sortie.

### Différents systèmes de dialogue

Une dimension supplémentaire de l'architecture d'un AC est le type de système de dialogue, "*une machine qui communique verbalement avec un utilisateur humain afin d'échanger des informations concernant un thème commun*" (Berg 2014). La nature de cette communication peut être caractérisée par (1) un système de contrôle sur un environnement (par exemple le contrôle d'une maison), (2) un système de question-réponse ou (3) un système de recherche d'informations.

Chatscript est principalement axé sur un système de question-réponse, bien que la recherche d'informations soit parfois possible grâce à la recherche de mots-clés via Wikipédia. NADIA répond à ces trois exigences.

## ChatScript

ChatScript (Wilcox and Wilcox 2017) est un outil permettant de créer des AC. Étant donné une entrée de l'utilisateur, les AC générés par ChatScript déterminent la réponse à retourner à lui retourner via

un ensemble de règles.

ChatScript semble être un choix adéquat en ce qui concerne l'objectif de ce travail. Ayant été récompensé du Prix Loebner (n.d.) à deux reprises, cette technologie s'inscrit dans l'état de l'art en ce qui concerne la création d'AC. Par ailleurs, la conception des règles ChatScript est déléguée au concepteur de l'AC, via des fichiers prévus à cet effet. Cela rend donc possible l'intégration de règles générées automatiquement.

## Le corpus Ubuntu

Le corpus Ubuntu (Liu et al. 2016) est un recueil de conversations écrites entretenues en temps réel sur le site d'Ubuntu (anglophone). Il s'agit d'un site communautaire dont le but est de rassembler des solutions aux problèmes techniques rencontrés sur le système d'exploitation Ubuntu. Chaque conversation de ce corpus est constituée d'échanges entre deux interlocuteurs uniquement.

## Objectifs

L'objectif de ce TER est de créer un modèle permettant de :

1. Généraliser les conversations humaines;
2. Générer des règles ChatScript à partir de cette généralisation.

Par "généraliser les conversations humaines", j'entends généraliser le sens de leurs énoncés. Autrement dit, modéliser un énoncé de façon à capturer son sens.

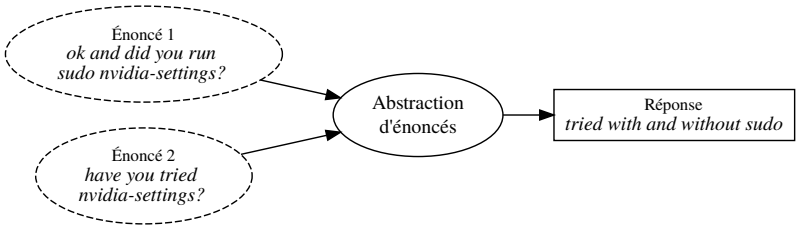


Figure 1: Généralisation d'énoncés partageant un sens similaire

Dans le contexte du corpus Ubuntu, l'idéal serait de créer un modèle généralisant les questions posées par les utilisateurs sollicitant un assistance technique. Autrement dit, étant donnée une question, celle-ci serait modélisée afin de pouvoir capturer toutes les questions ayant la même sémantique et de retourner une réponse unique, comme modélisé dans la figure 1.

Par ailleurs, généraliser les réponses à ces questions serait également optimal. Autrement dit, étant donnée une question, produire une réponse unique, pertinente et "propre" (correction de l'orthographe, de la syntaxe, etc.).

Dans le cadre de ce TER, pour raison de simplicité, nous ne faisons pas la distinction entre question et réponse. Nous généralisons donc uniquement certains énoncés (selon la méthode explicitée dans la section 2.3), sans nécessairement prendre en compte la proximité sémantique exigée par l'idéal. De manière alternative, nous considèrerons naïvement que le sens d'un énoncé est contenu parmi son lexique, notamment parmi les mots étant jugés comme "importants" (section 3.1).

Ainsi, l'objectif 1. est accompli par un processus de filtrage des termes pertinents dans un énoncé. Quant à l'objectif 2., nous procédons à la création d'ensembles indépendants de règles correspondant à un thème particulier.

La figure 2 illustre les résultats attendus des objectifs 1. et 2. Entre autres, l'objectif est de déterminer un ensemble de mots-clés dits "du domaine" afin de pouvoir généraliser l'entrée d'un utilisateur.

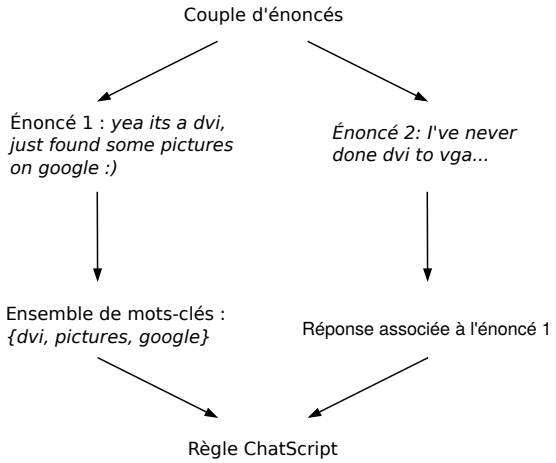


Figure 2: Modélisation du processus de génération de règles

Une brève introduction aux spécificités de ChatScript est dans un premier temps nécessaire à la compréhension de la démarche expérimentale (section 1). Dans un deuxième temps, nous présentons les détails cette approche expérimentale (section 2), laquelle est soumise à plusieurs paramètres que nous détaillons dans la section 3. Enfin, nous verrons en section 4 que l'évaluation d'AC constitue une problématique à part entière. Ainsi, après une proposition de méthodologie permettant d'évaluer les modèles obtenus de manière quantitative (section 4.2), nous procédons à une analyse quantitative d'erreurs pour raisons de simplicité (section 5).

# 1 Règles et thèmes

Cette section consiste à introduire les concepts clés et le fonctionnement global de ChatScript.

## 1.1 Les mots-clés des règles

Lors de la spécification des objectifs, nous avons supposé que le sens d'un énoncé était composé du sens de l'ensemble de ces mots-clés. Nous considérons un mot-clé comme un terme appartenant au domaine.

En ce qui concerne le corpus Ubuntu, le domaine au sens large est a priori l'informatique. Ainsi, comme illustré dans la figure 2, les mots "*dvi*" et "*google*" appartiennent au domaine de l'informatique, alors que "*some*" ("quelque") est bien plus général. Notons que "*pictures*" ne fait a priori par partie du domaine. En revanche, un jugement humain l'inclurait probablement du fait de la présence du mot "*google*". Nous n'avons pas l'ambition de capturer l'interdépendance des mots-clés dans cette étude.

Nous tenterons de déterminer l'appartenance d'un mot au domaine via les différentes techniques expliquées dans la section 3.1.

## 1.2 Le format des règles

Le rôle des règles est de déterminer une sortie en fonction de l'entrée émise par l'utilisateur humain. Celles-ci se présentent selon le format décrit dans la figure 3.

Voici les points essentiels à retenir :

- "u:" marque le début d'une règle et signale à l'AC que l'entrée peut être une affirmation ou une question. ChatScript offre d'autres symboles que nous n'utilisons pas ici.
- Les parenthèses "( ... )" contiennent les mots-clés associés à une règle.
- Les doubles chevrons "<< ... >>" indiquent que l'ordre des mots-clés dans l'entrée est négligé.
- Le reste de la règle (la partie suivant les parenthèses) correspond à la sortie retournée à l'utilisateur.
- Les commentaires (dénotés par "#!") serviront à indiquer un énoncé tel que se trouvant dans le corpus. Ceux-ci sont indiqués uniquement si l'information apportée est pertinente.

Notons que le format de règle choisi implique que tous les mots-clés d'une règle soient être présents en entrée (quel que soit leur ordre) pour que cette règle soit valide. La raison pour cela est expliquée dans la section subséquente.

```
#! commentaire
u: (<< mot-clé_1 mot-clé_2 ... mot-clé_n >>) Réponse émise par l'AC
```

Figure 3: Format d'une règle ChatScript

## 1.3 Généralisation d'un énoncé

Afin d'effectuer la généralisation d'un énoncé (comme expliqué dans l'introduction), nous procédons à un filtrage des mots-clés (section 3.1), puis nous négligeons l'ordre d'apparition de ces mots-clés dans cet énoncé. En négligeant l'ordre, nous nous assurons que plusieurs phrases contenant exactement les mêmes mots-clés activent la même règle. Ceci part de la supposition naïve que le sens d'une phrase est constitué de l'ensemble de ces mots-clés.

La négligence de l'ordre des mots est caractérisée par l'adoption des doubles chevrons dans chacune des règles.

## 1.4 Le format des thèmes

Chaque règle est répertoriée dans un fichier correspondant à un thème choisi par le concepteur de l'AC.

```
topic: ~nom_du_thème[mot_déclencheur_1 mot_déclencheur_2 ... mot_déclencheur_n ]
```

Figure 4: Format de l'entête d'un thème ChatScript

Un fichier thème commence par l'entête schématisé dans la figure 4. Les mots entre crochets correspondent à l'ensemble des mots déclencheurs qui servent à activer le thème. La présence d'un de ces mots dans une entrée d'utilisateur suffit à activer le thème.

### 1.5 Le mécanisme d'activation des thèmes et des règles

Un thème est activé en fonction des mots-clés se trouvant en entrée. Nous nous référerons spécifiquement aux mots-clés du thème par le terme "mots déclencheurs" car ceux-ci ont le rôle d'"activer" ce thème.

Il suffit de la détection d'un seul mot déclencheur pour qu'un thème soit activé. Une fois le thème activé, la première règle valide trouvée dans le fichier est activée.

## 2 Protocole expérimental

Afin de (1) généraliser les conversations humaines et (2) générer cet ensemble de règles, nous procédons par "prototypage". Autrement dit, la méthode consiste à produire un script de manière itérative - c'est-à-dire en remédiant à certains des problèmes rencontrés à l'itération $n-1$. Le script produit à chaque itération correspond à la procédure suivante :

1. Créer un corpus autour d'un thème commun.
2. Créer des couples d'énoncés.
3. Créer une règle pour chaque couple, les mots-clés étant obtenus à partir du premier énoncé et la sortie correspondant au deuxième.
4. Créer un ensemble de mots déclencheurs appropriés.
5. Créer un fichier thème ChatScript contenant l'ensemble des mots triggers et l'ensemble des règles obtenues.

### 2.1 Création d'un corpus autour d'un thème commun.

À titre d'exemple, le thème choisi pour cette expérience est le mot "*nvidia*" et constituera également le nom du thème. Plusieurs autres mots seront choisis afin d'évaluer les conflits entre les thèmes. Dans ce contexte, les autres thèmes seront précisés. Nous désignons chaque script résultant d'une itération comme "prototype". Chaque prototype est numéroté en fonction des approches adoptées.

### 2.2 Création des couples d'énoncés

Afin de générer des règles ChatScript, il convient donc de sélectionner des couples d'énoncés dans le corpus. Le premier énoncé de cet échange servira à extraire des mots-clés tandis que le second constituera la réponse retournée par l'AC.

Cette sélection n'est pas évidente. En effet, les échanges du corpus Ubuntu ne sont pas "homogènes", dans le sens où les interlocuteurs n'interagissent pas forcément tour à tour. Dans l'optique de créer des règles, le cas idéal serait tel que chaque interlocuteur interagit à son tour et que le nombre d'échanges soit pair. On obtiendrait ainsi $\frac{n}{2}$ règles pour $n$ énoncés dans une conversation donnée.

Cette réalité n'étant pas effective, la solution envisagée est de sélectionner uniquement les couples d'échange de type $AB$, où $A$ correspond à l'énoncé d'un interlocuteur et $B$ à l'énoncé de son homologue.

### 2.3 Création de règles à partir de couples

Une fois les couples d'échanges obtenus, la prochaine étape est de les transformer en règles. Pour procéder, on filtre le premier échange afin d'en obtenir les mots-clés et on retient le second en tant

que sortie. Afin de transformer ce premier échange en mots-clés, celui-ci subit un processus de filtrage afin de déterminer les mots-clés pertinents (section 3.1). Le deuxième échange nécessite quant à lui quelques traitements tels que la suppression des parenthèses et crochets (pour lesquels l'interpréteur retourne une erreur lorsque non fermés) ainsi que l'échappement des caractères spéciaux suivants :

ˆ - $ * @ : #

### 2.4 Création de l'ensemble des mots déclencheurs

Une fois les règles créées au sein d'un thème, il est nécessaire de déterminer un ensemble de mots déclencheurs de ce thème. Plusieurs stratégies sont mise en œvre afin d'y procéder (section 3.3).

## 3 Configuration de l'approche expérimentale

Afin de créer un modèle permettant de générer des règles automatiquement, j'adopterai l'approche décrite dans le protocole expérimental (section 2). Dans le but d'améliorer les résultats obtenus, ce protocole sera exécuté de manière itérative. À chaque itération, un prototype de script permettant de générer des règles sera créé et celui-ci sera soumis à une analyse d'erreurs (section 5). Nous tentons de résoudre ces erreurs puis d'appliquer les solutions de manière à produire un nouveau prototype dans l'itération suivante.

Le protocole expérimental varie ainsi en fonction de plusieurs paramètres ; le filtrage des mots-clés, la priorité des règles ainsi que la génération des mots déclencheurs. Nous tâcherons ici de présenter les différentes variantes de ces paramètres. Le résultat de ce travail consistera à déterminer quelle(s) combinaison(s) de ces paramètres produit le meilleur modèle de génération de règles.

### 3.1 Filtrage des mots-clés pour les règles

Déterminer des mots-clés pour les règles implique d'extraire ces mots-clés à partir d'énoncés du corpus. L'objectif de ce filtrage est de retenir uniquement les mots du domaine dans les énoncés concernés.

Il existe plusieurs formes de filtrage ; j'en définirai trois ici. La première, le filtrage positif, consiste à retenir uniquement les mots d'un dictionnaire spécifique. Une deuxième approche, à l'inverse, est d'inclure tous les mots ne faisant *pas* partie du dictionnaire. La troisième approche, et la plus complexe, est le filtrage en fonction du tf-idf, qui est un score déterminant l'importance de chaque mot au sein du corpus (section 3.1.3).

J'utiliserai ces techniques de filtrage dans le but d'obtenir les mots-clés les plus représentatifs du domaine.

#### 3.1.1 Filtrage positif par dictionnaire du domaine construit manuellement

La première approche consiste à filtrer positivement les mots appartenant à un dictionnaire du domaine. Ce dictionnaire a été construit manuellement à partir des mots sémantiquement les plus proches des termes “ubuntu”, “linux” et “computer”. Ces proximités correspondent à la distance cosinus de la représentation vectorielle de ces mots, obtenues avec *word2vec* (Mikolov et al. 2013) et implémenté avec *Tensorflow* (Abadi et al. 2016). On obtient un ensemble de 1778 mots qui constituent le dictionnaire.

### 3.1.2 Filtrage négatif par dictionnaire général

Cette approche consiste à retenir comme mot-clé tout terme ne figurant pas dans le dictionnaire choisi. Ce dictionnaire est constitué de la liste des mots de la langue anglaise librement distribuée par les systèmes d'exploitation Unix.

### 3.1.3 Filtrage en fonction du tf-idf

Le tf-idf s'appuie sur la loi de Zipf (Zipf 1935). Ce ratio est composé de la fréquence d'apparition d'un mot dans chaque document du corpus ("tf" pour "*term frequency*") et de la fréquence inverse de document qui quantifie l'importance du mot dans l'ensemble du corpus ("idf" pour "*inverse document frequency*").

Une fois que chaque mot du corpus obtient un score tf-idf, ceux-ci sont filtrés en fonction d'un seuil. Pour cette approche, j'ai choisi la moyenne des scores de chacun des mots comme seuil. Ainsi, chaque mot dont le score est supérieur à la moyenne est retenu comme mot-clé.

## 3.2 Priorité de position et d'activation des règles

Deux raisons justifient le besoin d'ordonner les règles dans un fichier thème. La première, étant donné que ChatScript sélectionne la première règle valide, est qu'il est essentiel de placer les règles les plus pertinentes en premier. La deuxième raison est que, en tant qu'être humain, le concepteur de l'AC éprouve probablement le besoin de visualiser les règles générées.

Nous considérons que la règle la plus cohérente à un énoncé est celle comprenant le plus grand nombre de mots-clés présents dans cet énoncé.

Les démarches présentées ici sont des tentatives d'ordonnancement des règles permettant à la fois d'activer les plus cohérentes et de les visualiser.

### 3.2.1 Par ordre alphabétique

La première démarche vise à trier les règles d'un thème par ordre alphabétique. Ainsi, celles-ci sont ordonnées en fonction de leurs mots-clés. Les ensembles de mots-clés les plus courts sont prioritaires sur les plus longs. Lorsque deux ensembles présentent le même $n^{\text{ème}}$ mot, le $n+1^{\text{ème}}$ mot détermine la priorité.

### 3.2.2 Par ordre inverse de longueur puis ordre alphabétique

La seconde démarche consiste à ordonner les règles par ordre inverse de longueur de l'ensemble des mots-clés d'une règle puis par ordre alphabétique.

## 3.3 Génération des mots déclencheurs

Les règles ChatScript sont regroupées dans des thèmes. Afin que l'AC puisse sélectionner un de ces thèmes, ceux-ci doivent contenir un ensemble de mots déclencheurs.

Lorsqu'un énoncé est entré par l'utilisateur, l'AC cherche dans un premier temps un thème approprié en fonction des mots de l'entrée. Dans un deuxième temps, une fois qu'un thème est trouvé, l'AC cherche la première règle valide. On remarque ainsi que les mots déclencheurs dépendent des mots-clés

des règles puisque ces premiers sont inutiles s'ils ne sont pas présents dans ces derniers. En effet, une règle ne sera jamais activée si le thème dans lequel elle se trouve n'est jamais activé.

Considérons un énoncé à $n$ mots, un thème $A$ et un thème $B$, qu'il existe dans le thème $A$ une règle dont les mots-clés couvrent $n$ mots de cet énoncé et qu'il existe une règle dans le thème $B$ dont les mots-clés couvrent $k < n$ mots de cet énoncé. Si le thème $B$ est activé, la règle du thème $A$ (qui est la plus cohérente) ne sera pas activée. Il s'agit donc de déterminer une approche permettant d'éviter les conflits entre les différents thèmes et d'activer le plus cohérent à l'énoncé.

#### 3.3.1 Union des mots-clés des règles

Une première approche naïve consiste à retenir l'union de l'ensemble des mots-clés de chaque règle du thème.

#### 3.3.2 Sélection des mot-clés au plus grand tf-idf

Nous verrons en section 5.3.1 que l'approche précédente présente certains inconvénients. Le filtrage par tf-idf ayant été implémenté, il semble logique que cette approche puisse également servir dans la sélection des mots déclencheurs.

L'approche vise simplement à retenir uniquement le mot-clé d'une règle ayant le plus grand tf-idf ; et donc le plus pertinent du domaine. L'ensemble des mots déclencheurs correspond ainsi à l'union de chacun des mots-clés retenus.

On note qu'il suffit qu'un seul mot-clé d'une règle appartienne aux mots déclencheurs pour que cette règle soit exécutable.

# 4 Méthodes d'analyses de l'expérimentation

Cette phase d'expérimentation implique l'évaluation d'un AC, ce qui constitue une problématique à part entière, à laquelle nous ne tenterons pas de répondre dans le cadre de ce TER. En effet, il semble difficile d'évaluer les AC produits par la démarche décrite. Cette section consiste ainsi à proposer des méthodes permettant d'évaluer les modèles générés.

## 4.1 Méthode d'analyse qualitative

Chez l'humain, la discrimination entre un énoncé émis par un autre humain et un énoncé produit par une machine relève de l'intuition. En conséquence, en partant du principe qu'un "bon" AC est tel que l'interaction avec celui-ci donne l'illusion d'interagir avec un humain, nous pouvons mesurer le ressenti d'une population face à un modèle donné.

Un système de dialogue naturel est une forme de système de dialogue dont l'objectif est d'améliorer l'utilisabilité et la satisfaction de l'utilisateur en imitant le comportement humain (Berg 2014). Cette même étude liste certains critères d'un système de dialogue naturel, comme l'usage de la langue naturelle, l'initiative mixte, la flexibilité et la coopérativité.

Ainsi, une forme d'évaluation serait de sélectionner un ensemble de couples d'énoncés et de les soumettre à une évaluation réalisée par un échantillon de population représentatif. Ceuxi-ci jugeraient les modèles selon les critères d'un système de dialogue naturel. Cette évaluation serait reproduite pour chacun des prototypes générés.

Cette approche présente bien entendu plusieurs inconvénients. Celle-ci serait longue et fastidieuse à mettre en oeuvre, étant donné qu'un échantillon de population représentatif et d'effectif adéquat serait requis.

Dans les limites de ce TER, nous procédons à une analyse qualitative d'erreurs biaisée, puisque l'auteur de l'expérience et de l'évaluation est le même. Cette analyse consiste à juger qualitativement les performances de chacun des paramètres du protocole expérimental en tant qu'analyse d'erreurs (section 5).

## 4.2 Méthode d'analyse quantitative

L'analyse qualitative auprès d'humains est coûteuse et limitée en termes de passage à l'échelle. De plus, celle-ci n'est pas toujours évidente à interprétée. Par contraste, l'analyse quantitative fournit une mesure objective et directement interprétable. La méthode pour procéder à une telle analyse n'est cependant pas intuitive en ce qui concerne l'évaluation d'AC. Après une brève synthèse des méthodes d'évaluation des systèmes de dialogue de l'état de l'art, nous nous intéresserons à une méthode spécifique d'évaluation de nos agents conversationnels.

### 4.2.1 Systèmes d'évaluation de systèmes de dialogue

Plusieurs travaux ont tenté d'automatiser le processus d'analyse quantitative avec différentes approches. La plus utilisée, BLEU (Papineni et al. 2002), est un système d'évaluation d'AC originellement conçu pour les systèmes de traduction automatique. Ce procédé fournit un score basé sur les cooccurences de *n-grams* de la réponse de référence ainsi que celle du système de dialogue évalué. Cependant, BLEU se base uniquement sur l'aspect lexical et néglige la proximité sémantique entre les énoncés. Il a par ailleurs été démontré que ce système fournit une évaluation faiblement corrélée avec le jugement humain (Liu et al. 2016).

Capturer cette proximité sémantique est l'une des motivations du système d'évaluation ADEM (*automatic dialogue evaluation model*), mis en avant par (Lowe et al. 2017), qui vise à mesurer la pertinence d'un énoncé de réponse en prenant en compte (1) le contexte du dialogue, (2) la réponse de référence du corpus et (3) la réponse générée par le modèle. Pour ce faire, ADEM se sert d'une analyse qualitative préalable associée à un corpus de conversations humaines. Un modèle est ensuite entraîné à partir des trois données énumérées, lequel cherche à minimiser la distance entre le score généré par ce même modèle et le score fourni par des humains pour chaque énoncé du jeu d'entraînement.

### 4.2.2 Méthode d'évaluation d'un agent conversationnel avec ADEM

Afin d'évaluer les AC créés à partir des différentes méthodes abordées, nous suggérons le processus décrit dans la figure 5, lequel a recours au système ADEM. La première étape consiste à collecter les couples d'énoncés de chaque dialogue et à les lier à ce même dialogue afin de pouvoir en retirer le contexte ultérieurement. Ces couples d'énoncés servent ensuite à générer un AC qui est ensuite évalué qualitativement (avec un score de pertinence) par des humains. On utilise par ailleurs les couples d'énoncé ainsi que chaque dialogue associé afin de calculer les vecteurs[1] de contexte du dialogue ($\mathbf{c}_i$), de la réponse de référence ($\mathbf{r}_i$) et de la réponse du modèle ($\hat{\mathbf{r}}_i$). Les scores ainsi que les vecteurs constituent ainsi le jeu de données final que l'on sépare en données d'entraînement et données de test. Après l'entraînement du modèle ADEM, on fournit les données de test à ce modèle pour obtenir l'évaluation finale de notre AC. Une fois le modèle obtenu jugé suffisamment fiable, il nous est ainsi possible de

[1] "plongements de documents" (*document embeddings*)

mesurer la qualité d'un AC similaire. Une moyenne des scores obtenus pour chaque échange permettrait de mesurer un système de dialogue de manière quantitative.

# 5 Analyse qualitative d'erreurs

Les différentes approches suivies ont mené à différents effets non désirables (ici appelés "problèmes"). Comme nous l'avons déjà mentionné, chaque problème rencontré à l'itération $n$ a abouti à une nouvelle approche implémentée dans l'itération $n + 1$. Le tableau 3 indique quels paramètres sont utilisés pour chaque prototype.

| Processus | Prototype 1 | Prototype 2 |
|---|---|---|
| Filtrage | Positif | Négatif |
| Mots déclencheurs | Union | Union |
| Priorité | Ordre alphabétique | longueur inverse |

Tableau 3: Les différents paramètres utilisés pour chaque prototype

| Processus | Prototype 3 | Prototype 4 |
|---|---|---|
| Filtrage | tf-idf | tf-idf |
| Mots déclencheurs | Union | max tf-idf |
| Priorité | longueur inverse | longueur inverse |

Pour chaque paramètre de l'approche, nous détaillerons ainsi les problèmes rencontrés.

## 5.1 Filtrage

La méthode de filtrage fait varier les résultats de manière significative. Les problèmes exprimés ici ne sont pas complètement issus du type de filtrage. Bien que le problème 1 (imprécision des mots-clés) perde de son amplitude avec l'adoption du filtrage négatif et du filtrage en fonction du tf-idf, ce problème reste néanmoins présent.

### 5.1.1 Filtrage positif

#### 5.1.1.1 Problème 1. Le filtrage des mots-clés des règles est imprécis

On observe que le filtrage positif avec dictionnaire du domaine génère des mots-clés peu représentatifs du sens des énoncés originels. En effet, le nombre de mots ainsi que le choix des mots du dictionnaire ne semble pas convenir pour un filtrage pertinent.

On remarque également que cette méthode de filtrage inefficace présente une conséquence majeure : l'existence de nombreuses règles concurrentes, c'est-à-dire que ces règles ont les mêmes mots-clés en entrée. Dans ce cas, l'AC ChatScript exécute la première de ces règles qui lui est présentée et ne peut qu'ignorer les autres ; puisque la première règle valide est activée.

Plusieurs solutions sont possibles pour ce problème : (1) rendre le dictionnaire plus exhaustif ou (2) opter pour une méthode alternative. Or, il semble qu'il existe des méthodes plus simples que la première. Dans l'approche 3.1.2, le filtrage négatif ainsi que le tf-idf sont mentionnés. Nous avons ainsi exploré

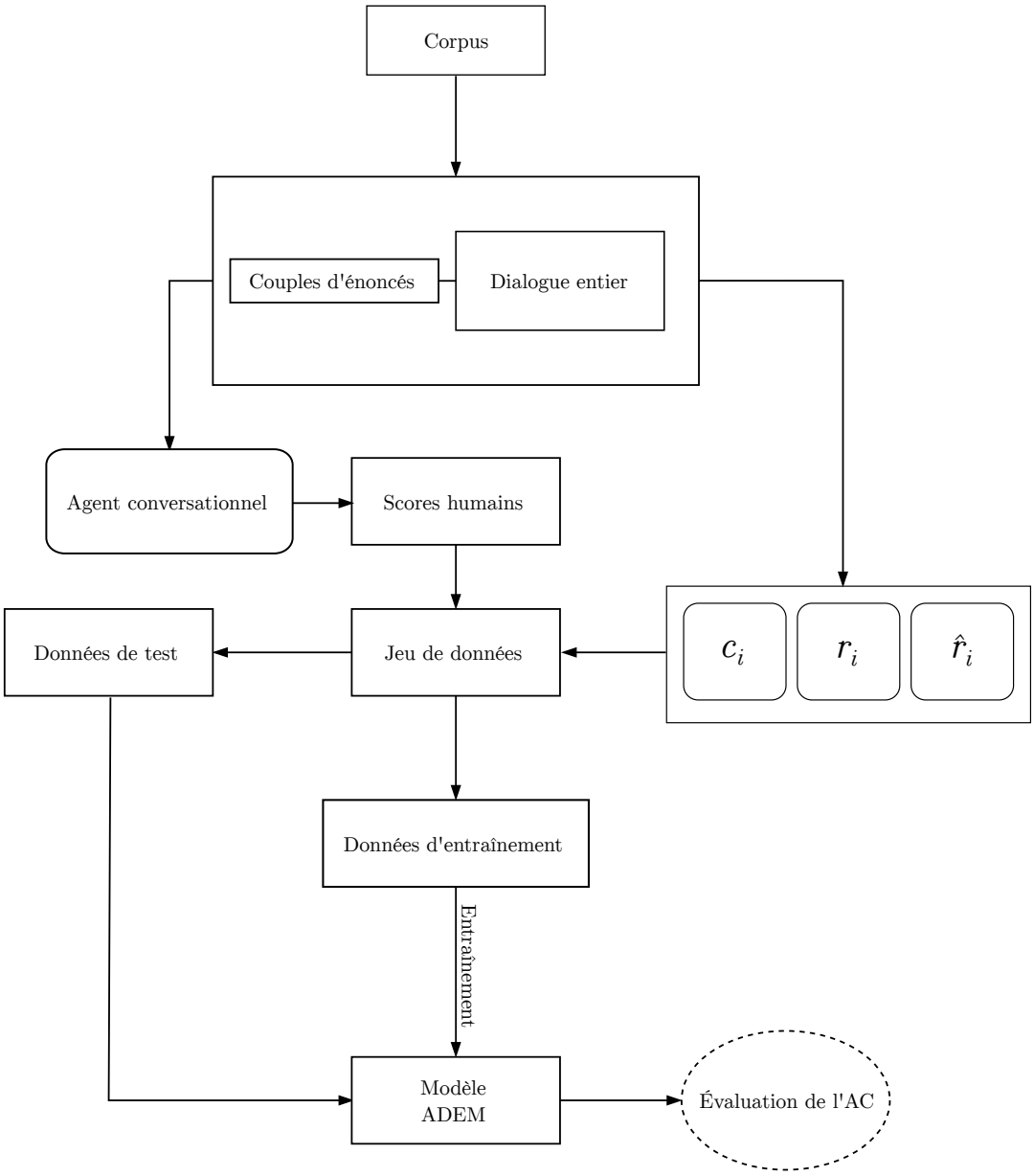


Figure 5: Processus d'évaluation d'un agent conversationnel avec ADEM

```
#! add in fstab but the umask sets read/write for mount /dev/hda1
/share ext3 umask=000 0 0
u: (<< add >>) I just copied the entry for my other drive basically
```

Figure 6: Exemple de filtrage positif avec dictionnaire du domaine

les deux méthodes afin de déterminer laquelle permet de produire des résultats plus pertinents. Un jugement qualitatif (mais biaisé) nous permet de conclure que le filtrage par tf-idf est plus pertinent.

## 5.2 Priorité d'application des règles

La priorité d'application des règles détermine la cohérence de la règle activée. Nous exposons ici les erreurs liées à cette priorité qui a pour conséquence de réduire la cohérence de la réponse retournée à l'utilisateur.

### 5.2.1 Par ordre alphabétique

#### 5.2.1.1 Problème 2. La règle la plus cohérente n'est pas toujours activée

L'ordre alphabétique des règles a un effet indésirable sur la performance de l'AC. Illustrons ce cas par un exemple. Étant donné l'entrée hypothétique : *"I've looked up on **google** which **specific** GPU driver I need for my **laptop** but couldn't find anything. The company doesn't provide any **support** for **linux** users", on attend que la règle de l'exemple 2 soit activée (figure 7). On observe cependant la sortie de l'exemple 1.

```
u: (<< google specific laptop linux support >>) how to do that?
```

(a) Exemple 1

```
u: (<< google >>) I've never done dvi to vga i am not sure if u
need to pass extra options...I know with dvi to hdmi setup one
for my 36" hdtv I had to pass extra options
```

(b) Exemple 2

Figure 7: Problème de priorité des règles : une règle moins pertinente est activée.

La raison pour cela est que l'ordre alphabétique place la deuxième règle (exemple 2) en premier. Ainsi, n'importe quelle règle comportant "*google*" dans ses mots-clés ne sera pas activée tant qu'il existe une telle règle la précédant.

Ce qui est intéressant est que l'ordre inverse ne donne pas le même résultat. En effet, si la première règle précède la deuxième, notre entrée hypothétique nous donne bien la sortie attendue. Ceci est dû au fait qu'une règle a besoin que tous ses mots-clés se trouvent dans l'entrée pour s'activer. De fait, l'ordre alphabétique "neutralise" toutes les autres règles suivantes partageant les mêmes mots-clés.

Cette observation nous mène ainsi vers une nouvelle approche :

1. Trier les règles par la longueur du nombre de leurs mots-clés (cela garantit la couverture maximale des mots-clés) ;
2. Puis trier par ordre alphabétique pour chaque longueur $n$.

Notons que le but de l'étape 2 est uniquement de pouvoir visualiser les règles dans un ordre logique. Cette nouvelle approche correspond à l'approche 3.2.2.

## 5.3 Génération des mots déclencheurs du thème

Les règles ChatScript sont regroupées dans des thèmes. Afin que l'AC puisse sélectionner un de ces thèmes, ceux-ci doivent contenir un ensemble de mots déclencheurs.

Lorsque l'utilisateur entre un énoncé, l'AC cherche dans un premier temps un thème approprié en fonction des mots de l'entrée. Dans un deuxième temps, une fois qu'un thème est trouvé, l'AC cherche la première règle correspondant à l'entrée. On remarque ainsi que les mots-déclencheurs dépendent des mots-clés des règles puisque ces premiers sont inutiles s'ils ne sont pas présents dans ces derniers.

Il s'agit donc de déterminer une approche appropriée de sélection des mots-clés depuis les règles afin d'éviter les conflits entre les différents thèmes.

### 5.3.1 Union des mots-clés des règles

#### 5.3.1.1 Problème 3. Concurrence entre les thèmes

Dans l'exemple du thème "*nvidia*", le script a produit un ensemble de 441 mots déclencheurs. Il convient que chaque thème comprenne des mots déclencheurs différents, sans quoi on peut s'attendre à des conflits d'activation des thèmes.

Le thème "*nvidia*" comprend entre autre les mots déclencheurs "*linux*", "*next*", "*pc*" et "*good*". Dans le contexte du corpus Ubuntu, ces mots sont très probablement utilisés de façon générale et des conflits au niveau des thèmes sont prévisibles.

De tels conflits ont en effet été observés. Plusieurs thèmes comprenant des mots déclencheurs similaires montrent un comportement imprévisible étant donné une même entrée de la part de l'utilisateur. Autrement dit, une règle d'un thème est tantôt activée tandis qu'une autre règle d'un autre thème est parfois activée.

Ce problème perd de son amplitude lorsque nous sélectionnons uniquement les mots clés au plus grand tf-idf de chaque règle comme composant des mots déclencheurs du thème. En revanche, le problème persiste toujours.

# 6 Conclusion et perspectives

Les démarches suivies dans le cadre de cette expérience ont permis de développer un système permettant de générer des règles automatiquement pour un AC ChatScript. Cet aspect de génération de règles, qui correspond à l'objectif 2., semble être fonctionnel.

En revanche, en ce qui concerne l'objectif 1., plusieurs problèmes relatifs au traitement automatique du langage naturel (TALN) émergent de notre approche. Premièrement, les règles générées retournent des réponses non pertinentes (lorsque l'on propose des énoncés différents de ceux du corpus d'origine), ce qui viole le critère d'initiative mixte. Ce manque de pertinence est dû à une trop forte dépendance aux données permettant de créer le modèle. Autrement dit, les énoncés d'entrée doivent correspondre à ceux d'origine pour que la réponse porte sens. Par ailleurs, il est fort plausible que l'hypothèse postulant que le sens d'un énoncé figure dans son lexique soit erronée. Ceci expliquerait ainsi la faible performance du modèle.

Afin de remédier à ce problème de surgénéralisation, une solution serait d'appliquer un modèle permettant de générer des "plongements de phrases" (*sentence embeddings*) pour chaque énoncé afin d'en capturer le contexte sémantique. Par la suite, nous pourrions procéder à un partitionnement de l'espace dimensionnel (*clustering*) afin de rassembler les énoncés similaires. Enfin, nous pourrions

envisager de sélectionner les mots les plus récurrents parmi les *clusters* en tant que mots-clés afin de générer des règles pour chacun d'eux. Bien que cette approche permette éventuellement d'améliorer la généralisation des premiers énoncés, celle-ci ne permet pas de généraliser les réponses à ces énoncés.

Deuxièmement, notre modèle est susceptible à des conflits entre les thèmes (problème 3), ce qui mène à des réponses imprévisibles. Ce problème n'émane pas d'un défaut du modèle en lui-même mais bien d'un choix lié au TALN. Il est probable que le choix supervisé des thèmes, c'est-à-dire choisir les thèmes de manière délibérée, ne soit pas adéquat. Encore une fois, un partitionnement des dialogues en ayant recours à des "plongements de documents" permettrait éventuellement d'obtenir des thèmes (implicites) de manière non-supervisée. Cette approche remédierait également au problème des thèmes à plusieurs mots et permettrait probablement de diminuer les chevauchements des mots déclencheurs et ainsi d'atténuer les conflits entre les thèmes. En outre, redéfinir la notion de thème semble primordiale à l'amélioration de notre approche.

En somme, l'aspect de génération de règles est opérationnel et ne présente pas de défaut apparent. Les problématiques liées au TALN nécessitent cependant une réflexion plus profonde afin d'augmenter la pertinence des réponses fournies par le modèle.

# Références


Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. "TensorFlow: A System for Large-Scale Machine Learning." 16:265–83.

Berg, Markus M. 2014. "Modelling of Natural Dialogues in the Context of Speech-Based Information and Control Systems."

———. 2015. "NADIA: A Simplified Approach Towards the Development of Natural Dialogue Systems." Springer, 144–50.

Lester, James, Karl Branting, and Bradford Mott. 2004. "Conversational Agents." *The Practical Handbook of Internet Computing*. New York: Chapman & Hall, 220–40.

Liu, Chia-Wei, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. "How Not to Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation." *arXiv Preprint arXiv:1603.08023*.

Lowe, Ryan, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. "Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses." *arXiv Preprint arXiv:1708.07149*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. "Distributed Representations of Words and Phrases and Their Compositionality," 3111–9.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. "BLEU: A Method for Automatic Evaluation of Machine Translation." In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–18. Association for Computational Linguistics.

Prix Loebner. n.d. "Prix Loebner." http://www.aisb.org.uk/events/loebner-prize.

Wilcox, Bruce, and Sue Wilcox. 2017. "Winning the Loebner's." https://github.com/bwilcox-1234/ChatScript/blob/master/WIKI/PAPERS/Paper%20-%20WinningTheLoebners.md.

Zipf, George Kingsley. 1935. "The Psycho-Biology of Language." Houghton, Mifflin.