# Interaction homme-machine en domaine large à l'aide du langage naturel : une amorce par mise en correspondance

Vincent Letard[1,2]
(1) LIMSI, CNRS, rue John von Neumann, 91405 Orsay cedex
(2) Université Paris-Sud, 91400 Orsay
letard@limsi.fr

**Résumé.** Cet article présente le problème de l'association entre énoncés en langage naturel exprimant des instructions opérationnelles et leurs expressions équivalentes et langage formel. Nous l'appliquons au cas du français et du langage R. Développer un assistant opérationnel apprenant, qui constitue notre objectif à long terme, requiert des moyens pour l'entraîner et l'évaluer, c'est-à-dire un système initial capable d'interagir avec l'utilisateur. Après avoir introduit la ligne directrice de ce travail, nous proposons un modèle pour représenter le problème et discutons de l'adéquation des méthodes par mise en correspondance, ou *mapping*, à notre tâche. Pour finir, nous montrons que, malgré des scores modestes, une approche simple semble suffisante pour amorcer un tel système interactif apprenant.

**Abstract.** We consider the problem of mapping natural language written utterances expressing operational instructions to formal language expressions, applied to French and the R programming language. Designing a learning operational assistant, which is our long term goal, requires the means to train and evaluate it, that is, a baseline system able to interact with the user. After presenting the guidelines of our work, we propose a model to represent the problem and discuss the fit of direct mapping methods to our task. Finally, we show that, while not resulting in excellent scores, a simple approach seems to be sufficient to bootstrap an interactive learning system.

**Mots-clés :** assistants interactifs, apprentissage artificiel, systèmes de question-réponse.

**Keywords:** interactive assistants, machine learning, question answering systems.

## 1 Introduction

Les avancées techniques et théoriques permettent d'effectuer des opérations de plus en plus puissantes et efficaces avec l'aide des ordinateurs. Pour autant, travailler avec la machine ne devient pas nécessairement plus simple. Exploiter la richesse de l'interaction homme-machine (Allen *et al.*, 2007; Volkova *et al.*, 2013) devrait permettre d'améliorer l'efficacité d'une tâche effectuée par l'humain à l'aide d'un ordinateur.

Notre objectif à long terme est de concevoir un assistant opérationnel dialogique apprenant par l'interaction à fournir une commande correcte en langage formel (LF) à partir d'un énoncé en langage naturel (LN). L'intérêt d'un tel système se trouve à la fois dans l'utilisation de l'outil informatique par un novice en programmation qui peut néanmoins exprimer son objectif en LN, et auprès d'utilisateurs plus avancés d'un langage de programmation, mais occasionnels ou bien pas toujours au plus haut niveau de complexité. Ainsi, l'utilisateur avancé pourrait enseigner au système les commandes complexes dont il se sert de loin en loin et les réutiliser facilement, et l'utilisateur novice pourrait profiter des connaissances dont dispose déjà le système (enseignées par d'autres utilisateurs au moyen de *crowdsourcing*[1] par exemple). Nous nous sommes intéressés pour ce travail au langage R. Il s'agit d'un langage de programmation pour l'analyse statistique et le traitement de données. Il est riche en fonctionnalités et souvent utilisé sous forme de scripts, qui permettent d'en oublier la syntaxe.

Avant tout, la conception d'un tel système apprenant requiert la collecte de données, et les premières tentatives ont souligné l'importance de l'utilisabilité pour le processus d'apprentissage. En effet, un système apprenant par l'interaction avec ses

[1]Le crowdsourcing est une méthode de collecte des données nécessaires à un processus avec l'aide d'un grand nombre de personnes. Ce processus peut centraliser ces données pour une utilisation locale, ou bien les réutiliser au profit de l'ensemble des personnes participantes.

| | Énoncés en LN | Commandes (en R) |
|---|---|---|
| 1 | Charge les données depuis **"res.csv"** | `var1 <- read.csv("res.csv")` |
| 2 | Trace un histogramme de la colonne **2** de **tab** | `plot(hist(tab[[2]]))` |
| 3 | Dessine la répartition de la colonne **3** de **tab** | `plot(hist(tab[[3]]))` |
| 4 | Somme les colonnes **3** et **4** de **tab** | `var2 <- tab[[3]] + tab[[4]]` |
| 5 | Somme les colonnes **3** et **4** de **tab** | `var3 <- sum(c(tab[[3]], tab[[4]]))` |

TAB. 1: Un échantillon d'associations entre énoncés en LN et commandes en LF. Ces exemples précisent la commande attendue pour chaque énoncé. Les éléments en gras sont liés avec les paramètres des commandes, cf. section 4.1. Les variables temporaires présentes dans la colonne de droite sont introduites par le système afin de permettre à l'utilisateur de faire référence à des résultats intermédiaires pour des traitement ultérieurs. Par ailleurs, on peut noter qu'un énoncé peut correspondre à plusieurs commandes différentes et inversement.

utilisateurs se doit de conserver leur intérêt sous peine d'obsolescence due à la perte de sa source d'information. Il nous faut donc fournir au système des capacités et connaissances initiales afin de permettre son développement incrémental avec l'aide des utilisateurs. Sans ces derniers, collecter des données se révèlerait bien plus fastidieux. Nous estimons que le prérequis minimal pour rendre le système utilisable est qu'il apporte de l'aide à l'utilisateur dans au moins 50% des cas. Ce premier seuil est volontairement bas car la visée d'un premier système est avant tout d'amorcer la collecte de données ; de meilleures performances devraient être atteintes par la suite grâce à ses nouvelles connaissances. Nous formulons en outre l'hypothèse que des méthodes simples de mise en correspondance entre LN et LF peuvent atteindre ce score.

Notre approche est basée sur une association directe paramétrée entre les énoncés en LN et les commandes en R. On utilise une base de connaissances $K$ composée d'associations paramétrées, telles que dans le tableau 1, pour sélectionner la meilleure commande à associer à l'énoncé-requête. Les énoncés $e^* \in K$, les plus proches de l'énoncé-requête selon une mesure de similarité $\sigma$, sont choisis. Les commandes associées $C(e^*)$ dans $K$ sont adaptées aux paramètres de l'énoncé-requête et une commande est retournée. Par exemple, étant donné l'énoncé-requête $e_{req}$ : "Charge le fichier data.csv", le système range les énoncés de $K$ par similarités décroissantes avec $e^*$. Pour le contenu de la base exemple dans le tableau 1, l'énoncé 1 doit intuitivement être classé premier, et le système doit retourner la commande : "`var1 <- read.csv("data.csv")`". Notons que plusieurs commandes peuvent être proposées en une fois afin d'offrir un choix d'alternatives à l'utilisateur.

Nous utilisons les mesures de similarité basées sur la distance de Jaccard, le tf-idf, et le score BLEU, et considérons différentes stratégies pour le choix des réponses retournées par le système. Les mesures de similarité évaluées se sont révélées être suffisemment complémentaires pour permettre l'utilisation de méthodes de combinaison, telles que le vote ou la classification automatique, pour améliorer *a posteriori* l'efficacité de la recherche.

La section 2 présente les domaines autour desquels s'articule notre problématique, ainsi que quelques travaux proches, nous posons ensuite notre formulation du problème et discutons de ses particularités en section 3. La section 4 détaille la méthode de constitution des associations et les différentes mesures de similarités utilisées, tandis que les paramètres d'analyse du LN et de l'ensemble de données utilisé sont exposés section 5. Enfin, nous donnons en section 6 les résultats d'expériences et discutons de leurs causes et implications.

# 2 État de l'art

## 2.1 Associer du langage naturel à du langage formel

Des problèmes proches ont été précédemment abordés à l'aide de différentes méthodes d'apprentissage. La transformation de requêtes en LN vers SQL est étudié par (Popescu *et al.*, 2003), le système requiert une grande précision dans ses réponses car l'objectif est de le rendre accessible au grand public et donc utilisable sur toutes sortes de bases de données. (Branavan *et al.*, 2009, 2010) utilise l'apprentissage par renforcement pour associer des instructions en anglais à des séquences de commandes en LF. Cela permet à l'association de prendre en compte les instructions haut-niveau et leurs constituantes. L'étendue des commandes élémentaires utilisables est cependant limitée aux possibilités de l'interaction graphique. Il en résulte que l'apprentissage ne peut pas produire de schémas très abstraits, du fait de la faible diversité des paramètres dans les commandes graphiques. Dans leur approche, (Kushman & Barzilay, 2013) abordent le problème de la génération d'expressions régulières correspondant à des descriptions en anglais à l'aide des grammaires combinatoires

catégorielles pour analyser le langage naturel et la représentation avec le $\lambda$-calcul pour inférer des règles de traduction du LN vers les expressions régulières. Cette approche générative par traduction permet la généralisation à partir des exemples d'apprentissage. Cependant, le pouvoir expressif des expressions régulières correspond aux grammaires de type 3 de la hierarchie de Chomsky. (Yu & Siskind, 2013) utilisent les modèles de Markov cachés établir par apprentissage une mise en correspondance entre des détections d'objets dans une séquence vidéo et des prédicats extraits de descriptions en LN. L'objectif de leur approche est différent du notre, mais le problème sous-jacent de trouver une association entre objets peut être comparé. Les objets appariés sont dans notre cas des expressions en LF plutôt que des détections dans une séquence vidéo.

### 2.2 Traduction automatique

La traduction automatique (Hutchins & Somers, 1992) renvoie habituellement à la transformation d'une phrase depuis un LN source en une autre portant le même sens dans un autre LN, appelé langage cible. Cette tâche est effectuée par la construction d'une représentation intermédiaire de la structure de la phrase a un niveau d'abstraction donné, puis la phase de génération encode l'objet obtenu dans le langage cible. Bien que suivant un objectif principal différent, l'une des tâches du projet XLike (Tadić *et al.*, 2012) était l'examen des possibilités de traduction d'instructions en LN (anglais) vers un LF (Cycl). Adapter une approche de ce type à un un langage formel cible opérationel (par opposition à Cycl qui est déclaratif) peut être une piste intéressante à étudier, mais il nous faut tout d'abord satisfaire l'objectif primaire de l'utilisabilité.

### 2.3 Recherche d'information

La question des systèmes de recherche d'information est comparable à celle de l'assistant opérationnel lors du parcours de sa base de connaissances. Les systèmes de question-réponse en particulier, révèlent des similarités avec l'assistant opérationel car les deux ont pour tâche de répondre à une expression en LN en recherchant la meilleure réponse dans l'ensemble des connaissances à leur disposition. Cependant, les systèmes de question-réponse s'appuient généralement sur la fouille de textes afin de trouver l'information correcte (Toney *et al.*, 2008). Cette méthode demande une grande quantité de données annotées (à la main ou par annotation automatique). Les tutoriels, cours ou manuels qui pourraient être utilisés à cette fin pour l'assistant opérationnel sont malheureusement trop hétérogènes et incluent des références complexes ou implicites à des connaissances générales (langage, algorithme, compilation). En un mot ils sont écrits pour l'humain, et il n'est pas envisageable de les utiliser en fouille de données sans une étude approfondie, quel que soit le mode d'annotation. D'où l'intérêt d'un assistant opérationnel apprenant capable de collecter des données standardisées et annotées à l'aide de l'utilisateur.

## 3 Formulation du problème

Ainsi que décrit en introduction, la base de connaissances $K$ est représentée par un ensemble d'exemples de la relation binaire $R : LN \rightarrow LF$ qui associe un énoncé en LN à une commande en LF. Si nous considérons le cas simple dans lequel la relation est fonctionnelle et injective, chaque énoncé est associé à une unique commande. Ce n'est pas réaliste car beaucoup d'énoncés en LN portent le même sens. Le cas d'une relation non injective couvre mieux les exemples usuels : chaque commande peut être associée à un énoncé ou plus, les exemples 2 et 3 du tableau 1 illustre cette situation. Cependant, le cas le plus réaliste est celui d'une relation qui n'est ni injective ni fonctionnelle. Plusieurs énoncés peuvent être associés à la même commande, et un seul énoncé ambigu peut renvoyer à plusieurs commandes différentes (voir les exemples 4 et 5 du tableau 1). Nous devons considérer toutes ces associations lors de la comparaison de l'énoncé-requête $e^*$ avec les énoncés de $K$ pour sélectionner une ou plusieurs commandes à retourner.

Pour cela, plusieurs stratégies non exclusives peuvent être considérées pour déterminer ce que le système retourne à l'aide de la mesure de similarité $\sigma : LN \times LN \rightarrow \mathbb{R}$ entre deux énoncés en LN. Typiquement, il faut déterminer si le système doit répondre, et si oui, combien de commandes il doit retourner.

La première stratégie est axée sur le nombre de réponses données pour chaque énoncé-requête $e_{req}$. Les $n$ premières commandes relativement au classement de leurs énoncés associés dans $K$ sont retournées. Étant donné $e_{req}$, le rang $r$ d'un énoncé $e \in K$ est donné par le nombre d'énoncés de $K$ dont la similarité avec $e_{req}$ est supérieure à celle de $e$ avec $e_{req}$.

$$r(e|e_{req}) = |\{e' \in K : \sigma(e_{req}, e') > \sigma(e_{req}, e)\}|$$

Le second choix de stratégie est de déterminer un seuil de similarité en dessous duquel les énoncés candidats de $K$ et leurs commandes associées sont considérés trop différents pour correspondre. Cela permet de choisir si une réponse est donnée ou non, et donc d'autoriser le silence du système, mais n'offre pas de contrôle sur le nombre de commandes retournées (bien qu'on puisse le conserver par la suite sous un seuil raisonnable). Cette stratégie retourne donc comme résultat l'ensemble des commandes dont l'énoncé associé dans $K$ a une valeur de similarité avec $e_{req}$ supérieur au seuil déterminé :

$$Res = \{c \in LF : (e, c) \in K, \sigma(e_{req}, e) > s\}$$

avec $s$ le seuil de similarité sélectionné. Le tableau 2 illustre le classement des énoncés de $K$ par similarité décroissante avec $e_{req}$. Notons que, bien que les énoncés de la base de connaissances aient été ordonnés, rien ne nous permet de discriminer les commandes qui leur sont associées. En conséquence, l'application de la première stratégie avec un nombre de commandes égal à 4 sélectionne les deux commandes associées à e1, et un sous-ensemble quelconque de taille 2 parmi les commandes associées à e4. Afin de rationaliser cette sélection, on peut envisager de pondérer chaque association de

| $e$ | $\sigma(e, e_{req})$ | commandes associées à $e$ dans $K$ |
|---|---|---|
| e1 | $0,80$ | {c2, c6} |
| e4 | $0,74$ | {c4, c5, c7} |
| e3 | $0,36$ | {c3, c6} |
| e2 | $0,36$ | {c1} |

TAB. 2: Exemple de classement pour un énoncé $e_{req}$ donné.

$K$ en fonction du nombre d'utilisations correctes et incorrectes (par exemple l'association e1 $\rightarrow$ c6 reçoit un poids de 4 car la commande c6 a été retournée pour e1 5 fois avec succès et 1 fois à tort). Ainsi, les commandes les "moins risquées" seront toujours privilégiées. Cependant, il ne s'agit que d'une optimisation de surface et elle demande des informations sur l'utilisation du système, ce dont nous ne disposons pas. D'autre part, on peut également remarquer qu'une commande peut apparaître à plusieurs endroits dans le classement des énoncés de $K$. c6 apparaît donc pour e1 et pour e3, avec deux valeurs de similarité différentes. Dans ce cas, seule la mieux classée sera retenue.

La formulation du problème proposée repose la fonction de similarité $\sigma$ et sur les associations présentes dans la base de connaissances. Nous allons maintenant présenter notre approche pour établir ces associations et les fonctions de similarité que nous y appliquons.

# 4 Approche

Nous avons initialement le résultat d'une analyse syntaxique simple de l'énoncé et de la commande. La première étape à effectuer est l'acquisition des exemples et la procédure de mise à jour de la base de connaissances. Nous examinons ensuite les méthodes pour rechercher une commande à partir de la base connaissances et d'un énoncé-requête donné.

## 4.1 Intégration dans les connaissances

L'association correcte entre énoncés et commandes requiert au moins la prise en compte de leurs paramètres respectifs (noms de variables, valeurs numériques et chaînes de caractères entre guillemets). Les représentations génériques des énoncés et des commandes sont construites à l'aide de l'identification des paramètres dans le couple à intégrer à la base de connaissances (voir tableau 1). Ces représentations sont utilisées par la suite pour reconstruire la commande à l'aide des paramètres de l'énoncé-requête.

La base de connaissances ne contient que les formes génériques des commandes. Il s'agit du texte de la commande comportant des références non résolues pour chaque paramètre qui a été associé à un élément de l'énoncé d'apprentissage. Ces références sont résolues à la phase de recherche par association avec les tokens correspondants de l'énoncé-requête.

## 4.2 Retrouver les commandes

Nous avons appliqué trois mesures de similarité différentes pour la recherche de commandes afin de comparer leurs points forts et leurs points faibles : l'indice de Jaccard, une aggrégation du tf-idf[2], ainsi que le score BLEU[3]. Le choix de ces trois méthodes est dicté par la diversité des caractéristiques qu'elles mesurent, qui provient elle-même des cadres dans lesquels elles ont chacune été développées.

### 4.2.1 Indice de Jaccard

L'indice de Jaccard mesure la similarité entre deux ensembles à valeurs dans le même domaine. Dans notre cas, nous comparons l'ensemble des mots de l'énoncé-requête en LN et celui de l'énoncé d'apprentissage pour la commande candidate. Ils sont chacun valués dans l'ensemble des tokens possibles. L'expression de l'indice de Jaccard adaptée pour deux énoncés $e_1$ et $e_2$ est :

$$J(e_1, e_2) = \frac{|M(e_1) \cap M(e_2)|}{|M(e_1) \cup M(e_2)|}$$

où $M(e)$ désigne l'ensemble des mots de l'énoncé $e$ utilisés. Il semble plus pertinent dans ce cas d'ignorer les mots outils. En effet, la comparaison des énoncés sous la forme d'ensembles de mots fait perdre toute information sur leur ordre dans la phrase. Sans le contexte, les mots outils n'apportent que peu d'information et introduisent plutôt un biais dans le ratio des nombres de tokens. L'indice de Jaccard est une méthode standard pour évaluer les co-occurrences des unigrammes. Elle devrait être plus efficace avec des données comprenant peu d'exemples ambigus en termes de vocabulaire.

### 4.2.2 tf-idf

La mesure tf-idf calcule une mesure de représentativité d'un mot dans un document, par rapport à un corpus. Elle permet donc, étant donné un mot, de classer les documents du corpus selon sa représentativité pour chacun d'eux. L'avantage de tf-idf est qu'il tient compte de la fréquence du mot dans le corpus. Ainsi, un mot apparaissant dans tous les documents, même s'il est très fréquent, ne sera représentatif d'aucun document. Il faut ici évaluer la représentativité d'une phrase plutôt que d'un seul mot. On utilise donc une aggrégation des valeurs de tf-idf pour chacun des mots composant l'énoncé-requête en LN.

$$tfidf_e(e_{req}, e_{comp}) = \frac{1}{|M\,(e_{req})|} \sum_{m \in M(e_{req})} tfidf(m, e_{comp}, E)$$

avec $E = \{enonce | (enonce, commande) \in K\}$ l'ensemble des énoncés de la base de connaissance, et où $e_{req}$ est l'énoncé-requête et $e_{comp}$, l'énoncé comparé. La fonction $tfidf$ elle-même s'exprime par :

$$tfidf(m, e_{comp}, E) = \frac{|\{m_e | m_e \in e_{comp} \wedge m_e = m\}|}{|\{m_e | m_e \in e_{comp}\}|} \times log\left(\frac{|E|}{|\{e \in E | m \in e\}|}\right)$$

Cette mesure ne nécessite plus la restriction aux mots pleins car tf-idf inclut déja la normalisation par rapport à la fréquence globale (dans le corpus) des termes.

### 4.2.3 Le score BLEU

Le score BLEU (Papineni *et al.*, 2002) est une méthode de calcul de similarité qui a été développée pour automatiser l'évaluation de la traduction automatique. Il présente une bonne corrélation avec l'évaluation par l'humain et est donc pertinent pour la recherche de paraphrases. Cett méthode s'appuie sur la mesure de co-occurrences entre $n$-grammes et permet de distinguer les énoncés candidats dans lesquels l'ordre des mots est trop différent de la celui dans la référence (énoncé-requête). La valeur de précision modifiée qu'elle introduit est basée sur le rapport des co-occurrences de $n$-grammes entre candidat et référence, sur la taille totale du candidat, normalisée selon $n$.

$$P_{BLEU} = \sum_{gram_n \in e_{req}} \frac{max_{e_{comp} \in E} occ(gram_n, e_{comp})}{|\{gram_n \in e_{req}\}|}$$

[2]Term frequency-inverse document frequency
[3]Bilingual evaluation understudy

où $occ(gram_n, e) = \sum_{gram'_n \in e}[gram_n = gram'_n]$ est le nombre d'occurrences du $n$-gramme $gram_n$ dans l'énoncé $e$. On peut noter que le dénominateur, qui correspond au nombre de $n$-grammes de l'énoncé $e_{req}$, peut aussi s'écrire $|e_{req}|-(n-1)$. La méthode de calcul du score BLEU comprend également une pénalité pour la brièveté, et ainsi empêcher les longs énoncés d'être trop désavantagés. Cependant, les énoncés comparés dans notre cas ne sont pas des documents de longueur très variables mais des expressions opérationnelles de tailles et d'amplitudes beaucoup plus raisonnables. Nous avons donc laissé de côté cette normalisation.

### 4.3 Filtrage des éléments syntaxiques

Avant d'évaluer leur similarité, nous avons appliqué plusieurs combinaisons de filtres sur les tokens des énoncés à prendre en compte. Il est possible de conserver ou non les mots outils ou les éléments non lexicaux. Ces derniers regroupent les valeurs numériques, les chaînes de caractères entre guillemets et les noms de variables. Les noms de variables sont les mots inconnus de l'analyseur syntaxique qui correspondent à une sous-partie identique dans la commande. Si les éléments non lexicaux sont conservés, ils sont transformés en substituts standardisés. De plus, il est possible d'appliquer ou non la lemmatisation des termes lexicaux. Par exemple, en ignorant les mots outils, en conservant les éléments non lexicaux et après application de la lemmatisation, le deuxième énoncé du tableau 1 deviendrait :

*Trace un histogramme de la colonne **2** de **tab*** → TRACER HISTOGRAMME COLONNE **XXVALXX XXVARXX**

## 5 Paramètres expérimentaux

### 5.1 Analyse des énoncés

Avant toute analyse lexicale, on recherche dans les énoncés en LN des expressions arithmétiques afin de les marquer en tant que telles. Les énoncés sont ensuite analysés à l'aide de WMATCH, un analyseur syntaxique générique à base de règles, développé par Olivier Galibert (Galibert, 2009). Il s'agit d'un système modulaire, disposant d'ensembles de règles pour le français et l'anglais. À titre d'exemple, voici à quoi ressemble le résultat d'analyse pour le premier énoncé du tableau 1 :

```
<_operation>
  <_action> charge|_~V </_action>
  <_det> les </_det>
  <_subs> données|_~N </_subs>
  <_prep> depuis </_prep>
  <_mot_inconnu>
    "res.csv"
  </_mot_inconnu>
</_operation>
```

Les mots marqués comme inconnus sont considérés comme des noms de variables potentiels, à rechercher dans la commande associée. Les chaînes de caractères comme `"res.txt"` sont également marquées afin de rechercher ultérieurement des correspondances possibles avec la commande. Nous posons une hypothèse pour simplifier l'analyse : les énoncés en LN sont supposés ne contenir aucune faute d'ortographe.

D'autre part, les commandes sont normalisées par la séparation de toute paire de caractères non liés sémantiquement. Les valeurs numériques ainsi que les noms de variables/fonctions sont identifiés et marqués comme paramètres de la commande.

Ayant défini la méthode d'analyse, nous allons à présent décrire les données utilisées pour notre expérimentation.

### 5.2 Constitution du corpus

Le corpus d'amorçage est composé de 605 associations entre 553 énoncés uniques en français et 240 commandes uniques en R. Le faible nombre de documents (tutoriels, manuels, documentations) différents qui décrivent une grande partie des commandes R ainsi que leur hétérogénéité rendent malheureusement irréalisable l'acquisition automatique d'exemples d'apprentissage sans une étude approfondie de leurs structures. En effet, ces documents sont écrits pour des lecteurs humains, disposant de références générales sur les tâches ; de la valeur de retour des fonctions usuelles aux principes généraux de la programmation, en passant par les problématiques usuelles d'utilisation des bibliothèques d'analyse statistique. C'est pourquoi chaque paire énoncé-commande de notre corpus a été ajoutée à la main, s'assurant que chaque élément reflète complètement l'information utile de l'exemple, et qu'ils respectent les spécifications. Celles-ci sont censées être le moins restrictives possible : un énoncé en LN doit simplement être formulé comme s'il s'agissait de demander l'exécution de la tâche R associée. Cela conduit donc à une majorité de phrases à l'impératif, qui reflètent, pour les personnes expérimentées, la manière dont elles exprimeraient les tâches concernées pour des non spécialistes.

### 5.3 Métriques d'évaluation

Les mesures permettant une évaluation pertinente du système dépendent de son objectif. Les valeurs de précision et de rappel pour les systèmes de question-réponse sont calculées comme suit :

$$P = \frac{\text{\# réponses correctes}}{\text{\# réponses données}} \qquad R = \frac{\text{\# réponses correctes}}{\text{\# réponses correctes dans } K}$$

Ces formules peuvent être appliquées pour un système retournant un seule réponse pour chaque énoncé. En effet dans le cas contraire, la précision mesurerait la proportion de bonnes réponses totale. Or, on s'intéresse plus précisément au nombre d'énoncés test pour lesquels une commande correcte a été retournée. Nous définissons la précision par énoncé $P_e$ et le rappel par énoncé $R_e$.

$$P_e = \frac{\text{\# énoncés corrects}}{\text{\# énoncés données}} \qquad R_e = \frac{\text{\# énoncés corrects}}{\text{\# énoncés corrects dans K}}$$

Le nombre d'énoncés corrects s'entend ici le nombre d'énoncés test pour lesquels le système a retourné au moins une commande correcte. On peut noter que la mesure de rappel n'est pas aussi pertinente ici qu'en recherche d'information : si l'on pose l'hypothèse que la situation montrée par les quatrième et cinquième associations du tableau 1 ne sont pas courantes[4], le nombre d'énoncés corrects pour un énoncé requête donné devrait être faible, et la plupart devraient être équivalents. Ainsi, l'objectif principal n'est pas de retourner le plus grand nombre d'énoncés corrects car un suffit. Nous écartons donc l'évaluation du système à l'aide du rappel.

Examinons à présent les scores obtenus pour la mesure de précision par énoncé selon les différents paramètres du système.

## 6 Résultats et analyse

### 6.1 Comparaison des mesures de similarité

Comme le montre le tableau 3 la mesure de similarité basée sur le tf-idf domine les deux autres mesures en comparaison individuelle, quelle que soit la combinaison de paramètres de filtrage. En effet, la forme des énoncés présents dans le corpus cause la répétition des mots du vocabulaire de l'interaction opérationnelle (verbes : "donner", "calculer", "afficher" ; noms : "vecteur", "matrice", "histogramme", ...). Cette structure peut expliquer que la fréquence inverse dans les documents (idf) soit particulièrement pertinente.

[4] L'augmentation du nombre de tâches différentes couvertes par le corpus rendra ces collisions plus fréquentes, mais cette hypothèse semble raisonnable pour une amorce.

| mots outils | inclus | | | | non inclus | | | |
|---|---|---|---|---|---|---|---|---|
| non lexicaux | inclus | | non inclus | | inclus | | non inclus | |
| lemmatisation | oui | non | oui | non | oui | non | oui | non |
| **Jaccard** | 38.5% | 34.6% | 21.2% | 23.1% | 36.5% | 36.5% | 21.2% | 23.0% |
| **tf-idf** | **50%** | **50%** | **36.5%** | **40.4%** | **48.0%** | **51.9%** | **36.5%** | **40.4%** |
| **BLEU** | 34.6% | 34.6% | 26.9% | 30.8% | 30.8% | 32.7% | 26.9% | 30.8% |
| **hasard** | 1.9% | | | | | | | |

TAB. 3: Mesures de précision par énoncé ($P_e$), en fournissant 3 réponses pour chaque énoncé requête test. L'ensemble des associations connues contient 85% du corpus, et l'ensemble de test 10%.

La lemmatisation et l'inclusion des mots outils ne semblent pas avoir une influence évidente sur la précision par énoncé. À l'inverse, on constate une amélioration dans tout les cas avec l'inclusion des éléments non lexicaux. Ce comportement provient au moins en partie de la longueur des énoncés de $K$ ($7,5$ mots en moyenne), dont certains sont assez courts pour ne contenir aucun token lexical significatif (*"In de A"*). L'analyse linguistique plus approfondie devrait permettre, notamment pour les mots outils, d'améliorer les performances par une exploitation plus fine de la structure des énoncés. Cependant, notre travail a pour objectif de préparer une amorce pour un système évolutif, et l'intérêt d'utiliser des règles d'analyse fixes est donc limité.

La figure 1a montre la précision obtenue avec tf-idf en fonction du nombre de commandes retournées pour chaque énoncé requête. D'après le graphique, il est intéressant de proposer au moins trois commandes à l'utilisateur afin d'obtenir une précision décente. En revanche, proposer à l'utilisateur un choix de plus de 5 éléments ne serait pas pertinent : le gain en précision par énoncé ne serait plus suffisant par rapport au temps supplémentaire nécessaire à l'utilisateur pour retrouver la bonne réponse (le cas échéant) parmi les propositions.

## 6.2 Autoriser le silence

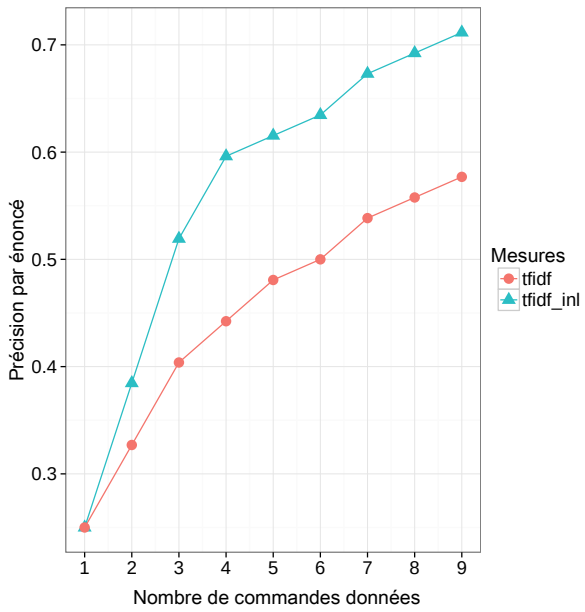
Afin d'autoriser le système à ne pas toujours donner une réponse, nous avons fixé un seuil absolu pour la valeur de similarité des commandes à retourner. Les résultats montrent, quelle que soit la mesure de similarité choisie, qu'au moins les 6 seuils les plus sélectifs donnent toujours lieu à des réponses fausses. Ce résultat peut être dû à l'existence dans l'ensemble de test, d'exemples non couverts par l'ensemble d'apprentissage. Plus vraisemblablement, il doit s'agir de commandes trop courtes (au plus un ou deux tokens lexicaux) pour être traitées correctement par la fonction de similarité.

## 6.3 Combinaisons

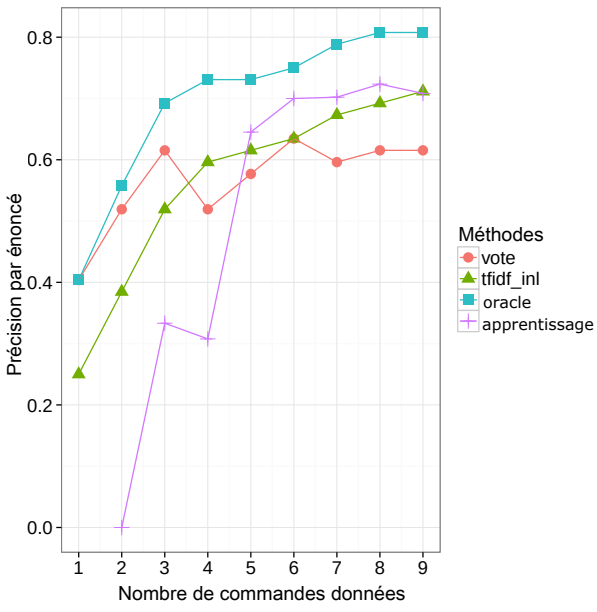
Une fois que chacune des méthodes proposées a été testée indépendamment, il peut être intéressant de tenter de les combiner. Cela permet notamment d'étudier leur complémentarité. L'oracle des 6 meilleures méthodes[5] montre une marge de progression intéressante (*cf.* figure 1b). Les résultats de la combinaison par le vote dépassent la meilleure méthode seule pour un nombre d'alternatives retournées inférieur à 4. Le vote atteint notamment 50% de bonnes réponses pour seulement 2 alternatives. La position de la courbe est moins claire pour un nombre d'alternatives plus élevé, mais notre problématique concerne l'utilisabilité du système, c'est pourquoi on s'intéresse plutôt aux gains pour un faible nombre de propositions. Cependant, il est indispensable d'effectuer les tests sur d'autres tirages de $K$ dans le corpus afin d'obtenir une mesure de la variabilité de ces résultats.

On peut également exploiter la complémentarité des méthodes par l'entraînement d'un modèle d'apprentissage artificiel pour combiner *a posteriori* leurs résultats. La régression logistique n'est pas directement applicable à notre cas puisque les listes de commandes retournées avec les différentes mesures de similarité sont discrètes ; c'est-à-dire qu'on ne peut pas les fusionner directement à l'aide d'un paramètre réel. Nous avons donc utilisé la classification afin d'identifier les schémas pour lesquels l'une des méthodes est meilleure que les autres. Il s'agit d'entraîner un système à reconnaître les méthodes auxquelles se fier en fonction des indices dont nous disposons. Les valeurs de similarité discrétisées ont été employées en tant qu'attributs et l'ensemble des mesures ayant donné la bonne réponse en tant qu'étiquettes de référence. Ces paramètres ont été testés à l'aide des machines à vecteur de support de `libsvm` (Chang & Lin, 2011) avec un noyau

[5] L'oracle donne toujours la bonne réponse si l'une au moins des méthodes considérées la fournit.




(a) Performances du système en utilisant la similarité tf-idf avec et sans **i**nclusion des tokens **n**on **l**exicaux.




(b) Comparaison des performances pour les différentes combinaisons de méthodes. Les 6 méthodes combinées sont tf-idf, Jaccard, BLEU, et ces 3 mêmes en **i**ncluant les tokens **n**on **l**exicaux.

FIG. 1: Précision par énoncé en fonction du nombre d'alternatives pour différentes méthodes. La précision par énoncé donne le nombre d'énoncés test pour lesquels au moins une des réponses proposées est correcte ; par cette définition, sa valeur ne peut donc qu'augmenter avec le nombre de réponses apportées pour chaque énoncé.

polynomial de degré 3 et à coefficient de degré 0 nul. Les résultats sont présentés sur la figure 1b. Comme on pouvait s'y attendre, l'entraînement sur le petit corpus dont nous disposons ne produit pas d'excellents résultats. La performance de la prédiction n'atteint pas 20%, et le modèle détermine seulement la distinction entre les cas où il faut choisir la meilleure méthode et les cas où il vaut mieux ne pas répondre (dû à l'étiquette de classe "aucune"). Les résultats des tests avec le modèle appris parviennent à dépasser légèrement tf-idf à partir de 5 réponses données, mais cela ne traduit pas une amélioration : d'une part, ce score est certainement obtenu grâce à l'abstention de réponse, qui est rendue possible par la présence de l'étiquette "aucune" lors de l'apprentissage, et d'autre part un tel nombre de réponses est déjà élevé si l'on considère que chacune des requêtes de l'utilisateur conduit à un choix multiple de 5 éléments.

## 7 Conclusion et perspectives

Nous avons appliqué 3 méthodes de calcul de similarité à la recherche de correspondances entre les énoncés en LN et les commandes en LF. Les résultats en termes de précision par énoncé ont atteint 60% de bonnes réponses après entraînement sur un petit corpus, tout en conservant le nombre d'alternatives de réponses proposées à l'utilisateur en deçà d'un seuil raisonnable ($< 5$ possibilités). Ces résultats valident l'hypothèse posée : les méthodes de mise en correspondance simple permettent d'atteindre un score de 50%.

Beaucoup de méthodes approfondies ou d'approches parallèles peuvent être considérées pour optimiser le score obtenu, comme l'ajout ou le développement de mesures de similarité plus adaptées (Achananuparp *et al.*, 2008), la combinaison de l'apprentissage artificiel et du vote, ou encore l'entraînement d'un modèle pour ordonner ou réordonner les listes d'énoncés similaires. Cependant, même si elles sont utilisables pour amorcer le système, ces méthodes généralisent peu et demandent de grands volumes de données pour être efficaces. De plus, la non fonctionnalité de la relation entre énoncés et commandes introduit des ambiguïtés impossible à résoudre à partir de la seule base de connaissances.

Grâce au système initial que nous avons élaboré, il maintenant possible de rassembler automatiquement plus de données en développant un assistant opérationnel apprenant. Ce dernier peut être vu comme une plateforme de *crowdsourcing* mais aussi, grâce aux performances acceptables que nous avons obtenues, comme un agent intelligent d'aide à la programmation. Nous développons actuellement une interface en ligne dans l'optique d'assurer ces deux objectifs. L'interaction

homme machine située devrait permettre la résolution en temps réel des ambiguïtés rencontrées, grâce à l'aide de l'utilisateur ou aux informations contextuelles du dialogue.

Néanmoins, l'application de l'interaction dialogique réelle pour l'utilisation de ce système pose de nouveaux problèmes liés à la complexité des commandes. Étant donné que toute commande sera apprise par le dialogue, il sera nécessaire de limiter leur longueur pour conserver un système utilisable. Une piste de travail est donc l'extension du modèle d'association, avec notamment la possibilité de référer à des sous-parties déjà connues, afin de permettre l'enseignement incrémental de commandes composées au système.

## Remerciements

Je souhaite remercier Sophie Rosset et Gabriel Illouz pour leurs relectures patientes et leurs encouragements, sans oublier Olivier Galibert pour ses excellentes remarques.

## Références

ACHANANUPARP P., HU X. & SHEN X. (2008). The evaluation of sentence similarity measures. In *Data Warehousing and Knowledge Discovery*, p. 305–316. Springer.

ALLEN J., CHAMBERS N., FERGUSON G., GALESCU L., JUNG H., SWIFT M. & TAYSOM W. (2007). Plow : A collaborative task learning agent. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, p. 1514 : Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999.

BRANAVAN S., CHEN H., ZETTLEMOYER L. S. & BARZILAY R. (2009). Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP : Volume 1-Volume 1*, p. 82–90 : Association for Computational Linguistics.

BRANAVAN S., ZETTLEMOYER L. S. & BARZILAY R. (2010). Reading between the lines : Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, p. 1268–1277 : Association for Computational Linguistics.

CHANG C.-C. & LIN C.-J. (2011). Libsvm : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, **2**(3), 27.

GALIBERT O. (2009). *Approches et méthodologies pour la réponse automatique à des questions adaptées à un cadre interactif en domaine ouvert*. PhD thesis, Université Paris Sud-Paris XI.

HUTCHINS W. J. & SOMERS H. L. (1992). *An introduction to machine translation*. Academic Press London.

KUSHMAN N. & BARZILAY R. (2013). Using semantic unification to generate regular expressions from natural language. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics* : North American Chapter of the Association for Computational Linguistics (NAACL).

PAPINENI K., ROUKOS S., WARD T. & ZHU W.-J. (2002). Bleu : a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, p. 311–318 : Association for Computational Linguistics.

POPESCU A.-M., ETZIONI O. & KAUTZ H. (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, p. 149–157 : ACM.

TADIĆ M., BEKAVAC B., AGIĆ Z., SREBAČIĆ M., BEROVIĆ D. & MERKLER D. (2012). *Early machine translation based semantic annotation prototype*. Rapport interne, XLike project, www.xlike.org.

TONEY D., ROSSET S., MAX A., GALIBERT O. & BILINSKI E. (2008). An evaluation of spoken and textual interaction in the ritel interactive question answering system. In *LREC*.

VOLKOVA S., CHOUDHURY P., QUIRK C., DOLAN B., REDMOND W. & ZETTLEMOYER L. (2013). Lightly supervised learning of procedural dialog systems.

YU H. & SISKIND J. M. (2013). Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, p. 53–63.