Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Récupération d'information dans un système de Question-Réponse à domaine fermé basé sur une ontologie en utilisant IBM Watson : une solution hybride à base de règles et d'apprentissage automatique

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Ingénierie Multilingue

par

Léon-Paul SCHAUB

Directeurs de mémoire :

Patrick Paroubek

Damien Nouvel

Encadrante :

Yulia Koloskova

Année universitaire 2017/2018

Résumé

Keywords :
System QA, IBM Watson, supervised machine learning, multi-classifier, hybrid system, *chatbot*, user interaction

Les systèmes de question-réponse pour un domaine fermé présentent l'avantage d'être faciles à implémenter, selon la littérature sur le sujet. En effet, les interactions étant limitées du point de vue du vocabulaire et des réponses possibles étant limitées, leur analyse et leur classification deviennent plus aisées. Cependant, par définition, leur base de connaissance est bornée et peut-être incomplète. À l'inverse, les systèmes à domaine ouvert se servent de nombreuses les sources existantes (le web par exemple) donc la taille de leur base de connaissance n'est pas finie. Ce travail porte sur l'analyse d'une question, indépendamment de son type, puis l'extraction des mots-clés qui permettent d'interroger une base de connaissance. Notre étude se situe dans le cadre d'un système de question-réponse dans le domaine administratif (secteur militaire), où une question est posée par un employé administratif. Ce dernier attend une réponse issue de la documentation réglementaire, qui sert de base de connaissances. Cette documentation est indexée dans un moteur de recherche sous forme d'une ontologie. Notre objectif est d'être capables d'analyser la question puis de fournir à l'employé la bonne réponse sous la forme d'un passage de la documentation. Pour réaliser cela, nous utilisons la technologie IBM Watson dont plusieurs services sont disponibles dans le commerce en libre-accès. Nous alimentons le système en questions-type, c'est-à-dire des questions dont nous connaissons par avance la thématique et la réponse dans la documentation, et qui nous sert de corpus de référence. Elles nous permettent d'entraîner un système de classification (à base de règles) et de prédire la thématique d'une question encore inconnue. Nous l'avons combiné avec une méthode d'apprentissage automatique multi-classifieurs, pour créer un système hybride, qui améliore le système à base de règles (de 77% à 80% de f-mesure). Chaque question posée par l'agent administratif conduit à la génération de passages-candidats issus de la documentation règlementaire. Nous tentons ensuite de déterminer le meilleur candidat à partir des critères extraits dans la question. Lorsque le système échoue dans l'analyse de la question, il interroge interactivement, au moyen d'un dialogue automatique directif (chatbot), l'utilisateur pour lui demander tous critères manquants, ou dont la représentation textuelle n'est pas reconnue par le système. Le système a été évalué et répond correctement à plus de 80% des questions qui lui sont posées.

Remerciements

Tout d'abord je tiens à remercier les membres de l'Er-Tim car si l'on m'a accordé le droit de présenter ce mémoire aujourd'hui c'est essentiellement grâce à leurs enseignements. Tous.
Ensuite je tiens à remercier mes camarades du master pour la patience dont ils ont su faire preuve pendant ces deux années, pour m'aider, me soutenir, le talent pour m'inspirer, et pour rendre cette formation agréable et plaisante, d'une façon ou d'une autre. Tous.
Enfin, je voudrais dire merci à mes collègues d'EVEA qui m'ont accueilli en son sein avec une sympathie et un respect émouvants. Chacun à leur façon. Tous.
Maintenant je voudrais remercier plus particulièrement ceux qui ont directement collaboré à rendre ce travail meilleur par la relecture notamment. A commencer par mes parents, excellents conseillers et brillants littéraires. Mais aussi tous ceux qui m'ont aidé à corriger que ce soient mes amis, mes collègues, ou les deux.
Au sein d'EVEA évidemment Yulia et Ikmel, mes deux compagnons de projet, qui prennent une place spéciale non parce que c'est avec eux que je passais le plus clair de mon temps en entreprise ni parce que leur talent et leur professionnalisme nous ont permis d'atteindre des résultats enviables dans nos projets, mais pour la confiance qu'ils ont toujours déposée en moi pendant ces quelques mois, sur mes idées, mes initiatives, nos désaccords, nos accords, nos réussites et nos revers. Cette expérience gratifiante en entreprise, c'est à eux que je la dois, je ne leur en remercierai jamais assez. Leur présence auprès de moi a été une inspiration en tout instant, je n'aurais jamais pu écrire ce mémoire sans notre travail d'équipe, ensemble.
Je finirai cette section par des remerciements plus personnels. Je voudrais remercier Damien pour son suivi dans mon mémoire, ses relectures professionnelles et ses conseils qui m'ont permis d'avancer. D'autre part, si j'ai acquis un intérêt pour l'apprentissage automatique et à l'aspect statistique et mathématique du traitement automatique des langues c'est en partie grâce à son enseignement
Bien sûr Patrick pour son aide incalculable tout au long de mon stage et de ma rédaction. Ses critiques quand je me trouvais dans le dur et ses encouragements quand je progressais m'ont éclairé comme une lanterne de bout en bout. Sa disponibilité et son investissement dans ce mémoire, au delà de l'honneur que cela me fait ressentir d'avoir été dirigé par Patrick, se reflètent dans sa construction car son aide m'a tout simplement permis de le rédiger tel qu'il est désormais. Enfin, nos réunions mensuelles m'ont donné des pistes de réflexion que je n'aurais même pas soupçonnées. Sa science et sa pédagogie ont été essentielles pour ma motivation à écrire ce travail. Sa sympathie et sa bienveillance ont été, elles, essentielles pour mon plaisir à écrire ce travail. Il s'en est sûrement aperçu, c'est lui qui m'a fait comprendre que je pouvais y arriver malgré mes doutes initiaux.
Je remercierai Mathilde ensuite, dont la patience et la gentillesse m'ont accompagné pendant toute notre formation, dans nos projets, et en dehors. Son soutien indéfectible en toute circonstance est quelque chose que je ne peux oublier et auquel je désire rendre hommage dans ce mémoire. Elle a été plus qu'une camarade, c'est une amie.
Pour finir, je voudrais remercier Chloé. J'aurais été tout à fait incapable de franchir un seul pallier en informatique si ce n'avait pas été par la confiance qu'elle avait déposée en moi dès le début de cette année. Elle a été un guide en toute circonstance. Vraiment merci.

TABLE DES MATIÈRES

Liste des figures

Liste des tableaux

INTRODUCTION

Grandes lignes du travail

Ce mémoire a été écrit en parallèle du stage de fin d'études dans l'entreprise EVEA COGNITIVE, un département d'EVEA, société d'informatique. Plusieurs objectifs ont été fixés pour ce stage. Une montée en compétence sur les techniques d'intelligence artificielle ainsi que sur les *chatbots*, un projet d'analyse et de redirection de flux de courriels (*mail dispatch* en anglais), et enfin un système de question-réponse qui constitue le coeur de ce mémoire.

Les systèmes de question-réponse appartiennent tant au domaine du Traitement Automatique des Langues qu'à celui de la recherche d'information. Le but d'un tel système est de répondre, par un fragment de texte existant ou une génération automatique de texte, à toute question posée EN LANGAGE NATUREL par un utilisateur.
Voici son fonctionnement : la question posée en langage naturelle est analysée par des annotateurs basés sur des règles ou de l'apprentissage automatique. Puis elle est indexée dans une base de données et des **mots-clés** en sont extraits.
Nous en trouvons deux sortes : le **sujet** ou **thématique** c'est-à-dire ce sur quoi porte la question.
Exemple :*"'Qui est le successeur du président Hollande "'* Le sujet de la question est **Hollande**. Nous pouvons déjà apprécier un premier niveau de complexité : le système doit comprendre qu'il s'agit de François Hollande et non pas du pays. Il doit donc utiliser le contexte, composé ici de *'Qui' 'successeur'* et *'président'*. C'est la seconde sorte de mot-clé, ce que l'on appelle des **indices** ou **critères**. Cet ensemble de **sujet** et d'**indices** sert à formuler des hypothèses sur les passages-candidats.

La partie la plus importante consiste à déterminer lesquels semblent les candidats les plus fiables. Pour ce faire, le système cherche des preuves : réponses à des questions avec sujet similaire, distance textuelle entre la question et un passage dans le document, pertinence des indices donc hiérarchisation de leur importance..., pour pouvoir attribuer un score à chaque candidat et sélectionner celui qui s'est vu octroyer le plus haut score pour finalement en extraire la réponse.

Ces systèmes sont assez anciens dans le domaine informatique. Le premier [**?**] appelé **BASEBALL** permettait à un utilisateur de connaître des résultats de matches de baseball. Plus tard, [**?**] mettent au point le système **LUNAR** pour interroger une base sur des données en astronomie, et dont les résultats ont atteint 90% de bonnes réponses pour des évaluateurs humains ne connaissant pas à l'avance les données d'apprentissage.

Depuis une vingtaine d'années et la montée en puissance des moteurs de re-

cherche tels que (1998) Google ou plus récemment (2013) Qwant en France, le système de Question-Réponse est considéré comme l'étape supérieure à l'indexation opérée par le moteur, car il ne se contente plus de fournir une liste de documents mais il est capable de choisir lequel sera le plus pertinent pour une question posée.
A partir de 1999, le **TREC** (Text REtrieval Conference) [Martin et al., 2001], conçu comme une compétition pour encourager les travaux sur les systèmes de recherche d'information, a permis aux systèmes de question-réponse de se développer en leur procurant un atelier et des méthodes d'évaluation. En France, le projet **EQUER** (Evaluation de Question Reponse), fonctionnant de façon similaire au TREC, [Ayache et al., 2006] voit le jour. Le meilleur système pour le français obtient 70% de réponses correctes.

Avec l'augmentation exponentielle des données du Web, la plupart des recherches se portent désormais sur les questions à domaine ouvert (ou multi-domaine) dont un des pionniers est START [Katz et al., 2006].

Pour notre étude, nous nous situons dans un domaine plus restreint. Autrement dit, nous possédons une base structurée, donc exhaustive, dans laquelle notre moteur de recherche tente de trouver la réponse à une question posée en langage naturel.
La base de connaissance est une documentation administrative dans le domaine du militaire concernant des problématiques de ressources humaines.

Il nous semble important de définir ici quelques notions lexicales spécifiques à ce mémoire.
Un *end-user* : traduit de l'anglais par 'utilisateur final' C'est le client. La personne pour qui le système de question-réponse est mis au point. Ici, c'est un personnel militaire.
Un *agent* : c'est l'intermédiaire humain entre le end-user et la base de connaissance. Ici, c'est un personnel administratif.
Une *thématique* : sujet (ou topic) d'une question. La recherche de la réponse se base sur la thématique. Fait partie de l'ensemble des mots-clés.
Un *critère* : ou indice. Représentation linguistique d'un concept syntaxique ou sémantique permettant de filtrer les réponses du moteur de recherche. Fait également partie de l'ensemble des mots-clés.
Les questions sont posées par des agents qui, ne possédant pas une connaissance suffisante de la règlementation, nécessitent une précision officielle par rapport à un cas concret venant d'un end-user.
Pour cela, on nous a fourni des questions-type, c'est-à-dire des exemples de questions créés artificiellement censés représenter les problèmes rencontrés habituellement par les end-users. En reliant chacune de ces questions-types à un paragraphe dans la documentation règlementaire, nous sommes parvenus à construire un corpus de référence. Nous parlons plus en détail des jeux de données dans le chapitre 3.

Nous essayons d'identifier la thématique d'une question et de la corréler à une des questions-types afin de donner la réponse attendue. Cela pose plusieurs problèmes auxquels nous allons tenter d'apporter des éléments de réponse à travers ce mémoire.
D'abord se pose celui de la complétude des questions-types : **couvrent-elles vraiment suffisamment de types de questions que peuvent poser les agents ?**

Sont-elles assez générales pour que nos annotateurs parviennent à calquer sur elles une question inconnue ?
Ensuite, se pose le problème de la reconnaissance des critères dans la réalisation textuelle de la question : **que faire si la question est mal posée, ou si le vocabulaire n'est pas reconnu ? Si nos règles ne couvrent pas toutes les variantes ou les anomalies orthographiques possibles ? Ou s'il manque des informations dans la question pour la relier à une question-type, voire à une thématique générale ? Dans quelle mesure sommes-nous capables de recomposer et de compléter la question en interceptant les critères implicites ou manquants de façon à ce que le moteur de recherche possède toute l'information récupérable et soir ainsi capable de proposer une réponse avec un taux de confiance élevé ?**
Cela nous conduit vers la réflexion principale : **comment savoir si une question contient des éléments manquants, ou si l'utilisateur ne nécessite, par exemple, qu'une réponse générique ? Quels procédés doivent être mis en place pour résoudre ce problème ?**

Pour y parvenir, nous utilisons un système de question-réponse. Plus précisément, nous nous inspirons de DeepQA, le système Question-Réponse d'IBM, appelé plus communément Watson, qui a gagné le jeu *Jeopardy !*, nous utilisons également la plate-forme Bluemix (aujourd'hui IBM Cloud) disponible en libre service, et Watson Explorer, un logiciel en distribution privée permettant de réaliser de la fouille de texte. Nous exploitons ses ressources et son architecture UIMA (Unstructured Information Management Architecture) [Ferrucci and Lally, 2004].
Pour extraire les possibles critères manquants dans une question, nous implémentons un agent conversationnel (*chatbot*) grâce au service Watson Conversation, qui sert à la fois d'enveloppe (Wrapper) pour les analyses de contenu détectant la thématique d'une question, et à la fois d'agent actif en demandant à l'utilisateur des précisions sur son problème.
Par exemple, si l'agent reçoit en question :"Un stagiaire sera-t-il remboursé de ses déplacements pendant son stage s'il rentre chez lui tous les quinze jours "? Le *chatbot* comprend qu'il s'agit d'une **demande de remboursement** (critère) pendant un **stage** (critère) lors d'un **retour intermédiaire** (thématique).
Il lui manque dans ce cas ci une information sur la durée du stage et une autre sur le justificatif de transport, car un militaire n'est remboursé que si le stage dure au moins un mois, et s'il présente ses factures de transport.
Le *chatbot* pose alors des questions directives et basiques avec une syntaxe ergonomique invitant l'utilisateur à répondre par des mots ou des petits groupes de mots faciles à interpréter :"Le stage du militaire dure-t-il **PLUS** ou **MOINS** de quatre semaines ?" cela incite l'utilisateur à répondre par un des mots en capitale.
Lorsque toutes les informations ont pu être correctement récupérées, nous pouvons enfin proposer plusieurs réponses à l'utilisateur. Elles sont classées par le moteur de recherche Watson Explorer (WEX) grâce à un taux de confiance calculé par l'algorithme de régression logistique (apprentissage automatique)

Pour l'analyse des questions, nous développons un système de classification hybride qui permet d'obtenir de meilleurs résultats dans l'évaluation que celle uniquement basée sur des règles. Dans la mesure où notre corpus est composé de couples question-réponse, nous essayons de diviser ces couples en deux dimensions

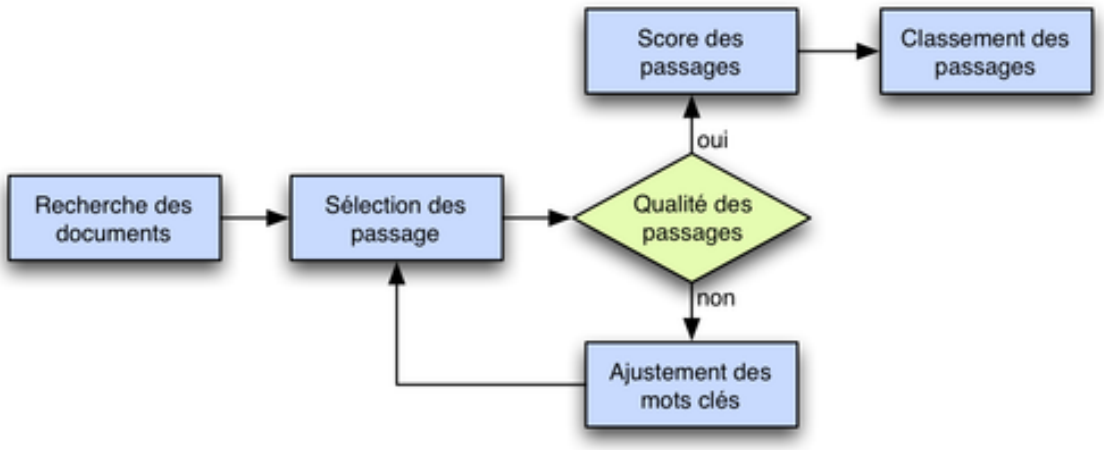


FIGURE 0.1 – schéma d'un système de question-réponse

distinctes, que nous exploitons par différentes méthodes d'apprentissage automatique et en utilisant les réseaux de neurones artificiels. Le chapitre sur les résultats détaille les performances de toutes nos expérimentations, et tente d'apporter un éclairage sur chaque performance.

Première partie

Contexte général

CHAPITRE 1

ETAT DE L'ART

Sommaire

Nous ne pouvons pas expliquer notre chaîne de traitement sans détailler ce qui a déjà été réalisé jusqu'à maintenant dans le domaine des systèmes de Question-Réponse. En effet, différentes approches existent et différentes méthodes sont appliquées. Pourtant, à l'heure actuelle, peu de systèmes sont commercialisables, surtout dans le domaine ouvert.

1.1 Tour d'Horizon

Selon [Kolomiyets and Moens, 2011], la différence entre domaine ouvert et domaine fermé repose dans le fait que ce dernier est destiné à des utilisateurs d'un champ bien spécifique et qui s'appuie sur une base de connaissances construite manuellement donc demeurant plus couteuse. A l'inverse, le système à domaine ouvert s'attèle à répondre à toute question quel que soit son thème. Il est donc plus difficile à implémenter.
Dans l'exploitation de données structurées, il existe également une opposition entre l'extraction d'information pure et l'analyse sémantique, comme démontré par [Yao et al., 2014], cette dernière servant dans le cas où l'extraction d'information traditionnelle échoue à récupérer une réponse correcte par la trop grande disparité sémantique entre la question et les passages candidats à contenir la réponse.
Les travaux se servant de l'analyse sémantique deviennent actuellement les plus utilisés comme AquaLog [**?**] ou bien le STAGG de [Yih et al., 2015] qui demeure encore aujourd'hui l'état de l'art pour les systèmes de question-réponse basés sur ontologie. Nous pouvons également citer [Nguyen et al., 2009] pour le vietnamien, YodaQA (utilisant un système hybride) [Baudis and Sedivy, 2015] pour le tchèque, qui par ailleurs s'inspire de la solution DeepQA d'IBM Watson qui, elle, représente l'état de l'art dans les systèmes hybrides [Gliozzo et al., 2013], dont nous parlerons en détail dans le chapitre suivant. Il existe aussi des boosteurs de moteur de recherche hybrides ontologie-web comme FALCON [Harabagiu et al., 2000]. Microsoft a également fait des recherches sur l'analyse sémantique pour les relations uniques question-réponse [Yih et al., 2014].

Pour la désambigüisation des dialogues, nous trouvons Querix [Kaufmann et al., 2006] toujours basé sur les ontologies.

Dans le domaine ouvert, le nombre de travaux a explosé depuis deux ans avec l'avancée massive dans la puissance de calcul des processeurs et le développement des *chatbot*s. Un des premiers systèmes à domaine ouvert s'appelle AnswerBus [Zheng, 2002]. C'est un système semblable à PageRank [Page et al., 1999] basé sur le Web, pour répondre aux questions. Il possédait un très bon score pour l'époque au TREC (0.7 de f-mesure) et possédait en plus la particularité d'être multilingue : nous pouvions poser des questions jusque dans six langues différentes. La réponse, elle, demeurait en anglais.
Les recherches utilisent désormais, en plus du TREC comme méthode d'évaluation, le Standford Question-Answering Dataset (SQuAD) [Rajpurkar et al., 2016]. . C'est un corpus disponible en ligne contenant plus de cent mille couples de question-réponse issus de Wikipédia. Aujourd'hui, l'état de l'art se situe autour de 85% de F-mesure. La F-mesure est une métrique calculant la moyenne harmonique entre la précision du système et sa capacité à retourner le maximum de bons résultats possibles. Ce sont les réseaux de neurones artificiels qui fonctionnent le mieux dans ces systèmes comme la co-attention dynamique [Xiong et al., 2016] et [Bearman, 2017] avec une dimension temporelle. Microsoft Asia [**?**] et MEMEN[Pan et al., 2017] utilisent également des réseaux de neurones artificiels. [Kumar et al., 2015] on créé un système de réseaux dynamiques de mémoire permettant de corréler l'entrée d'un utilisateur avec toutes les itérations précédentes. Il permet donc la résolution d'anaphores.

Il existe aussi des systèmes uniquement basés sur l'analyse syntaxique de la question et des documents candidats avec les arbres de dépendance [Katz *et* Lin, 2003] [Litkowski, 1999]. Certaines systèmes utilisent la distance d'édition sur les arbres [Punyakanok et al., 2004], Le chevauchement entre les relations de dépendance avec le système PiQUASso [**?**]. Enfin le raisonnement à travers les variations syntaxiques [**?**].
Sorti en 2009, Wolphram Alpha, un système de question-réponse construit sur une gigantesque base de connaissance est aujourd'hui très utilisé des agents conversationnels, par exemple dans SIRI.
Enfin JAVELIN [Nyberg et al., 2002] demeure le seul système hybride ontologie-syntaxe profonde à l'heure actuelle.

1.2 Travaux similaires

Notre étude, cependant, porte, comme nous le mentionnions précédemment, sur un domaine fermé (ou restreint), par conséquent les méthodes, même proches, diffèrent quelque peu. Par exemple, nous sommes contraints de posséder une base de connaissance suffisamment bien structurée pour que le moteur de recherche fonctionne correctement.
Pour le domaine médical par exemple, MedQA [Yu et al., 2007] donnait de meilleures réponses que les systèmes de l'époque (google, pubmed). Nous trouvons aussi HONQA [**?**] réputé pour ses réponses claires et précises, EAGil [Gobeill et al., 2015] qui transmet un mot-clé comme réponse et AskHermes [Cao et al., 2011] considéré comme le plus efficace car le plus rapide des trois. Olelo [Neves et al., 2017], un système dynamique toujours dans le domaine médical a des résultats finaux proches

de l'état de l'art.
Les travaux français dans le domaine médical sont avancés également. Le Système Esculape [Embarek, 2008] utilise l'extraction de relation pour prédire la bonne réponse. Le projet PatientGenesys [Zweigenbaum *et al*,2016] qui sert d'entraînement pour un médecin humain sur un patient virtuel pour l'évaluation des diagnostics obtient de très bons résultats même si la campagne d'évaluation n'est pas achevée.
Pour la langue chinoise, [Guo,2008] a développé un système hybride ontologie-web sémantique pour domaine fermé. De même, [Ferrandez *et al.*,2009] ont construit le système QUACID, souple, qui permet d'intégrer des ontologies de divers domaines restreints.
Pour l'arabe, [Hammo et al., 2002] entraîne le système QARAB avec un corpus tiré d'articles de journaux.
Dans le domaine éducatif, nous pouvons citer [Derici et al., 2015] qui se sert de modèles de markov cachés pour analyser les questions de son système.
Dans le domaine géographique enfin, nous pouvons citer [Luque, 2006], un système exclusivement oral pour répondre à des questions de géographie espagnole.

1.3 Les *chatbots*

Nous connaissons tous les *chatbots* récents comme SIRI (Apple, 2011) Alexa (Amazon, 2014) et Cortana (Microsoft, 2014). Ils sont très récents. La technologie de l'agent conversationnel ne date pourtant pas d'hier. Le premier système fonctionnant sur des règles appelé **ELIZA** [Weizenbaum, 1966]. Bien que très imparfait, c'est le premier système qui a essayé sérieusement de passer (sans succès) le test de Turing [Tur,]. Puis, PARRY [Kenneth Colby, 1971] un *chatbot* censé déceler la schizophrénie chez des patients.
Aujourd'hui, le *chatbot* reste peu performant : il est par exemple incapable de gérer une coréférence allant au delà de la phrase précédente : "Quelle question t'ai-je posée au début ?"
Certains travaux récents cependant tentent de trouver une solution à cette problématique en s'appuyant notamment sur l'apprentissage profond (deep learning) et les réseaux de neurones artificiels comme [Vinyals and Le, 2015] qui cherche à créer une conversation avec un réseau de neurones en s'appuyant sur des sous-titres de films. [Shao, 2017] tentent de créer une conversation où les dialogues de chaque personne sont longs mais où la qualité doit rester régulière.
Pour la résolution des coréférences, [Shao, 2017] et [Wiseman et al., 2016] apportent des solutions à base de réseaux de neurones récurrents.

1.4 Conclusion

La littérature se montre bien plus riche dans cette discipline depuis une vingtaine d'années avec l'apparition du web sémantique et le développement des ontologies. En effet, les travaux pour la plupart traitent le problème des questions-réponses sous l'angle du domaine ouvert, et des réseaux d'information. Malgré cela, il n'existe encore que peu de systèmes opérationnels dans l'industrie, de nos jours. Ceux qui fonctionnent le mieux sont ceux se servant de *chatbot*s, facilitant ainsi l'interaction homme-machine pour d'une part récolter des informations pour entraîner le système et d'autre part évaluer ce même système. Ces derniers ont explosé sur le marché et dans les publications scientifiques depuis quatre ans.

CHAPITRE 2

WATSON

Sommaire

Dans ce chapitre, nous présentons l'intelligence artificielle Watson, nous parlons de sa création, de ses prouesses, de la façon dont elle a été industrialisée et de sa médiatisation jusqu'à aujourd'hui.
Ensuite, nous expliquons l'architecture de son système de question-réponse.
Enfin nous terminons ce chapitre par expliquer deux applications particulières de Watson que nous avons utilisées dans notre travail, un outil de création de règles linguistiques, et un outil d'apprentissage automatique.

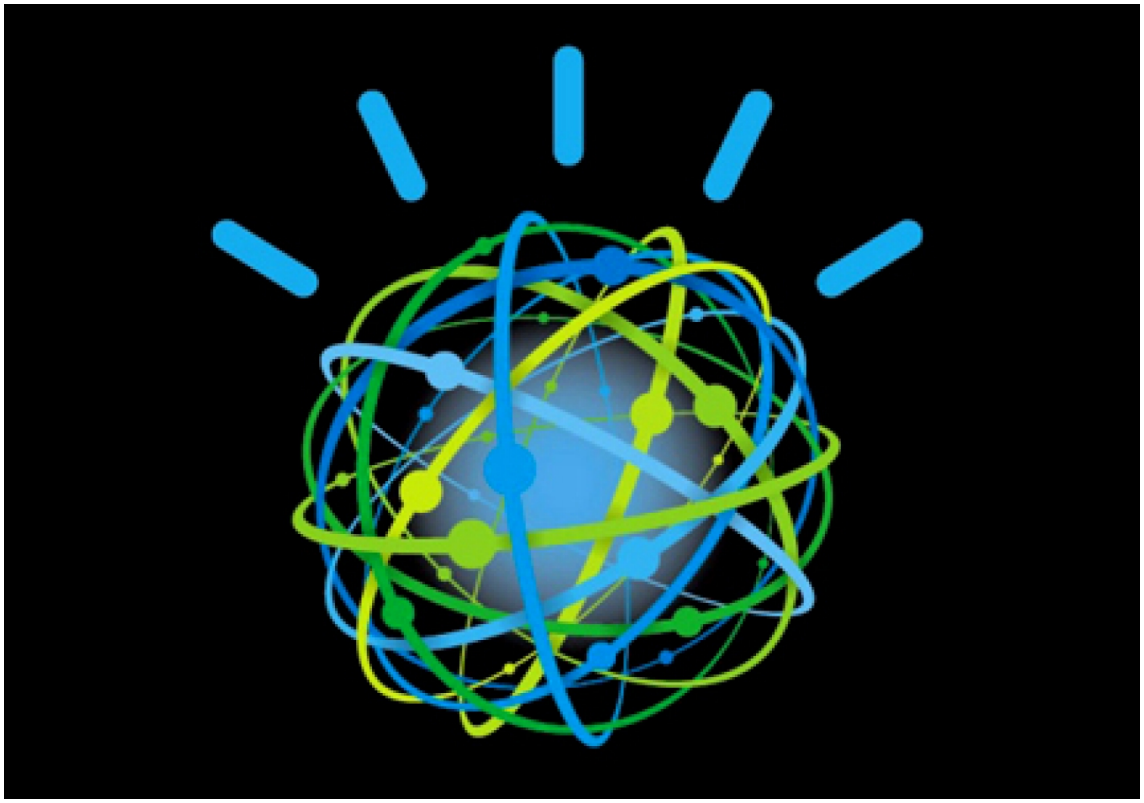


FIGURE 2.1 – logo de Watson

2.1 Watson hier et aujourd'hui

2.1.1 Description

Commençons par un peu d'étymologie : le terme Watson ne vient pas du nom du docteur, collègue et ami du célèbre détective, mais du créateur d'IBM (International Business Machines), Thomas J. Watson.

C'est un programme informatique d'intelligence artificielle conçu pour répondre à des questions posées, écrites ou orales, en langage naturel. Il peut de même y répondre à l'oral et à l'écrit, ce qui sous-entend l'utilisation de plusieurs technologies : la reconnaissance vocale et la retranscription de l'écrit, connus sous les noms de 'Speech To Text' et 'Text To Speech'. Son plus grand exploit reste à ce jour sa victoire dans le célèbre jeu télévisé américain *Jeopardy !*.

Watson fonctionne avec l'environnement de développement libre Apache UIMA (Unstructured Information Management Architecture) créé par IBM, et entre autres par le chef de projet de Watson, David Ferrucci. UIMA, écrit en Java et en C++, peut se définir comme une application capable d'analyser et d'explorer une grande quantité d'informations non structurées telle que des corpus de texte, mais aussi des données audio-visuelles. Il possède un langage de requêtes permettant d'interroger une base indexée où sont structurées en *facettes* les données d'entrée.

Une *facette* dans l'univers de Watson est une classe syntaxique ou sémantique permettant d'analyser un texte. Le contenu de chaque facette réunit toutes les représentations linguistiques (instances) appartenant à celle-ci. Grâce à ces facettes, il est possible de réaliser des requêtes ensemblistes, de comparer deux facettes pour les corréler ou non, d'explorer l'évolution d'une facette ou d'un ensemble de facettes dans le temps (opération appelée **écart**) ou de façon historique par rapport à l'époque précédente sélectionnée (opération appelée **tendance**).Chaque facette peut contenir des sous-facettes, elles-mêmes comparables avec des facettes de n'importe quel niveau. Par exemple, une catégorie grammaticale (part-of-speech en anglais que l'on abrège en POS) est une facette. Un adjectif en est une sous-facette, le qualificatif est lui-même une sous-facette de l'adjectif et ainsi de suite. Autre exemple, dans une analyse d'intention de courriels selon leur corps (par opposition à l'objet du mail jugé moins représentatif du but de l'expéditeur du mail), nous créons une facette **RELANCE**, dans laquelle nous fournissons comme exemple toutes les représentations textuelles du concept de relance (Ex : "je reviens vers vous, je vous ai déjà contacté pour.., c'est la troisième fois que.." ...).
Nous déclinons ces représentations : je/nous ; troisième/nième... En utilisant, entre autres, des expressions régulières pour formaliser en grammaires formelles ces exemples concrets.

Une expression régulière est un formalisme mathématique pour représenter un automate à états finis qui correspond à une classe de langage particulière dans la classification de Noam Chomsky.
Par exemple, le paradigme du verbe **chanter** à l'indicatif présent peut se décrire par l'expression régulière suivante : *"chant(e(nt|z)?|ons)"* qui s'interprète ainsi :
la chaîne de caractère *'chant'* suivie impérativement de : soit le caractère *'e'* lui-même suivi de façon optionnelle (l'option est symbolisée par *'?'*) par *'nt'* ou (l'union est symbolisée symbolisée par *'|'*) *'z'*, soit la chaîne de caractère *'ons'*.
Nous avons ainsi décrit une grammaire formelle avec la totalité (6) des formes possibles **correctes** du présent de l'indicatif de "chanter".

L'architecture UIM permet de décomposer les applications internes d'un système en briques manipulables : la détection de la langue, la segmentation spécifique du texte selon celle-ci, la détection d'entités nommées, la lemmatisation.. indispensables à la création des facettes par défaut, à leur tour nécessaires pour personnaliser des facettes sémantiques.
Nous pouvons créer des facettes, utilisant celles pré-construites dans l'architecture, afin de créer des règles.
Dans la mesure où le projet est libre et ouvert, plusieurs annotateurs créés par la communauté sont disponibles en ligne et intégrables à une chaîne de traitement. Par ailleurs, Watson contient par défaut plusieurs annotateurs profonds activables comme par exemple la détection de l'émotion selon le contenu textuel, ou la tonalité de la voix.
Afin que le système de question-réponse s'améliore, Watson est alimenté par des bases de connaissance et des ontologies, en particulier par DBPedia (aujourd'hui remplacé par WikiData), WordNet et Yago.
En définitive, Watson est un ensemble de technologies regroupant un système de

FIGURE 2.2 – le jour où Watson rivalisait avec le champion en titre

question-réponse que nous spécifions dans la section sur Jeopardy !, mais il utilise aussi la technologie du *chatbot*, de la reconnaissance vocale et de l'analyse de tonalité de la voie.

2.1.2 Génèse

En 1997, le super-calculateur d'IBM DeepBlue [Feng-hsiung Hsu *et al*, 2002] bat le champion d'échecs de l'époque, le Russe Gary Kasparov. Alors que des dizaines de publications voient le jour suite à cet exploit, c'est en 2004 que surgit l'idée chez IBM de développer un système informatique capable de concurrencer le cerveau humain dans un système de question-réponse à domaine semi-ouvert.
En 2006, David Ferrucci prend les commandes du projet DeepQA. Ce dernier a pour but d'implémenter un système de question-réponse basé sur ontologie suffisamment puissant pour rivaliser avec des adversaires humains en terme de précision et de rapidité dans le jeu télévisé *Jeopardy !* et ainsi de démontrer la puissance de l'intelligence artificielle. Les premiers essais sont très décevants et IBM doute de la capacité de l'équipe de Ferrucci à relever le défi *Jeopardy !* Eux estiment qu'il leur faut entre quatre et six ans pour remporter le jeu.

2.1.3 Le défi *Jeopardy !*...

Jeopardy ! est un jeu télévisé créé en 1964 et encore de nos jours diffusé aux Etats-Unis. Le principe est le suivant : "**À partir de réponses communément appelés des indices(*clues* en anglais), trois candidats doivent trouver la question correspondante. Chaque bonne réponse (c'est-à-dire chaque bonne question) rapporte une somme d'argent, chaque erreur la fait**

perdre. Ils peuvent choisir entre six catégories et cinq valeurs d'indices par catégorie."*(source Wikipedia)*. Chaque catégorie se voit attribuer cinq questions posées successivement et classées en ordre croissant par rapport à leur valeur (lors de la première manche, la première vaut 500$ et la dernière 2000$)
Par exemple : le présentateur propose plusieurs thèmes tels que géographie, football, cinéma.., le premier candidat en choisit un, disons le cinéma, et le présentateur donne les indices, sous forme de réponse :*"Ce contrebandier tombe amoureux d'une princesse, en 1980, malgré ses deux pains aux raisins à la place de cheveux"* Le candidat doit poser la question qui correspond aux indices fournit :*"Qui est Han Solo ?"*
A travers cet exemple, nous pouvons déjà relever plusieurs difficultés. Tout d'abord, nous y trouvons beaucoup d'informations, le nom commun exclusif à un être animé, au singulier masculin (contrebandier) sujet du verbe d'action également exclusif à un être animé (tomber amoureux), donc il s'agit d'une personne, et d'un homme. Ensuite, nous apercevons un cadre spatio-temporel : 1980. Après, nous observons un argument (au sens grammatical) intéressant : on nous parle d'une princesse, argument fort du verbe principal.. Les indices mentionnées précédemment enrichissent les recherches de départ et permettent dans un premier temps d'effectuer une pré-sélection des candidats susceptibles d'être la bonne réponse.
La dernière information, l'expression *malgré ses deux pains aux raisins à la place de cheveux*, faisant référence à la célèbre coupe de cheveux, demeure la partie la plus difficile à exploiter, même pour un humain, pour plusieurs raisons : la première c'est qu'il faut comprendre la phrase, ensuite il faut l'interpréter, et enfin la remettre dans son contexte, c'est-à-dire le film et le personnage auxquels elle fait référence.
Watson doit opérer dans un premier temps une analyse morpho-syntaxique de la phrase pour repérer le sujet (thématique ou *topic* en anglais) et les indices (critères, ou *clues* en anglais).
Dans un deuxième temps, il doit d'une part être capable de générer des hypothèses qui deviendront des potentiels passages-candidats, et d'autre part être capable d'aller chercher dans les sources les preuves (*evidences* en anglais) qui confirment ou infirment ces hypothèses.
Par exemple, il tente de faire des combinaisons 'contrebandier + 1980 ; contrebandier + princesse ; princesse + 1980 + pain aux raisins'... dans sa base de connaissance, et récupère ainsi des preuves que telle ou telle réponse de la base semble la plus plausible.
Après avoir réuni les preuves, Watson a besoin de donner une note (*score* en anglais) aux passages-candidats. Si plusieurs candidats concordent sur un même thème de réponse, ces candidats sont susceptibles d'obtenir de meilleures notes.
La quatrième étape consiste à générer un texte avec la réponse extraite du passage-candidat choisi.
Enfin, Watson doit être rapide, très rapide. En effet, *Jeopardy !* étant un jeu à 'buzzer', la précision du candidat reste inutile s'il ne 'buzze' pas assez vite. Autrement dit, Watson doit être capable d'effectuer toutes ces tâches en moins de trois secondes.
Nous allons tenter de résumer le plus succinctement possible le travail de David Ferrucci et de son équipe ainsi que l'architecture de DeepQA. Le paragraphe suivant est un schéma résumant l'article ***Building Watson : An Overview of the DeepQA Project***

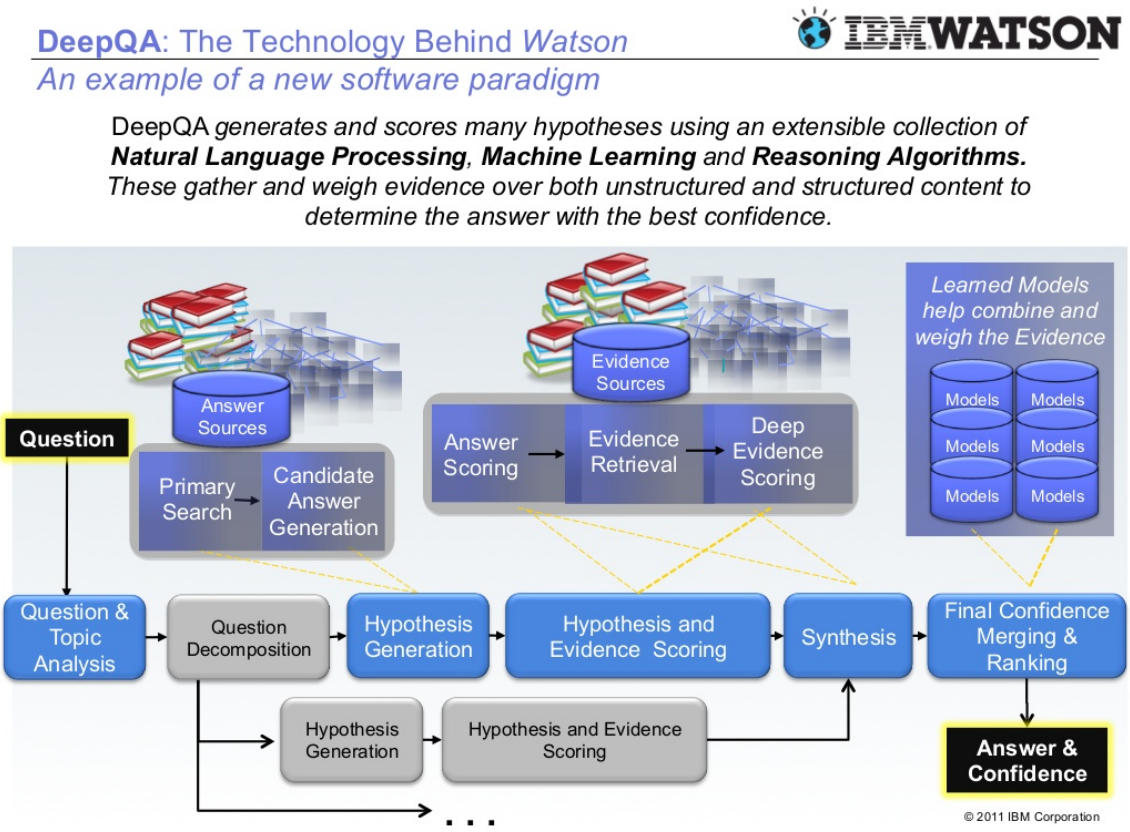


FIGURE 2.3 – schéma (en anglais) de DeepQA

1. Récupération du contenu de la question : Système hybride (règles, apprentissage automatique)
   a) Analyse des questions d'exemple (corpus de référence)
   b) Beaucoup de sources (dbpedia, wordnet, yago)
   c) Expansion du corpus de référence
      i. Identifier le sujet des documents et extraire des contenus similaires depuis le web.
      ii. Extraction de 'text nuggets' (explication d'un concept lexical par une phrase ou un syntagme, dans le même paragraphe voire la même phrase que le concept lui-même : fonction métalinguistique dans le schéma de Jackobson)
      iii. Noter ces nuggets en fonction de la quantité d'information qu'elles possèdent par rapport au document d'origine.
      iv. Rajouter la nugget la plus informative au corpus étendu.
2. Analyse des questions
   a) Déterminer comment la question sera traitée par le reste du système
   b) Classification des questions :
      i. Identifier des types (puzzle, factoïde, maths, définition..) des questions, puis en fonction effectuer un traitement particulier.
   c) 'Focal' Détecter des LATs (lexical answer type)

i. Déterminer si un candidat est une instance de la LAT, cela implique un risque élevé d'erreurs (contre-sens, ironie, utilisation erronée des sujets..)
ii. Permet de d'utiliser indépendamment plusieurs algorithmes de Question-Réponse.
iii. Focal : partie de la question qui, remplacée par un candidat, devient auto-suffisante. Souvent elle contient de l'information pour la réponse. Grammaticalement, elle est presque toujours SUJET ou OBJET de l'indice (importance de l'analyse syntaxique de profondeur pour la détection des arguments).

d) Détection des relations
i. Syntaxiques ou sémantiques
ii. Du focal jusqu'au score de la réponse
iii. La détection des relations dans un domaine plus vaste reste faible. Nécessité de cibler correctement le sujet.

e) Décomposition
i. Basé sur des règles et l'apprentissage automatique (LDA)
ii. Hypothèse : l'interprétation correcte aura un meilleur score après la collecte de preuves et utilisation d'algorithmes
iii. Améliore d'une façon générale la confiance en la réponse

3. Génération d'hypothèses
a) « Recherche primaire »
i. Quantité (le plus de candidats possibles)
ii. 85% de rappel
iii. Utilisation (Indri et Lucene) pour l'indexation et le moteur de recherche (requêtes SPARQL)
iv. Plusieurs requêtes par question
v. Triplets basés sur les entités nommées (topic) de la question
vi. Pour un nombre réduit de LATs qui ont été identifiés comme des LATs similaires, la réponse-candidat peut être générée à partir d'une liste fixe dans un stock d'instances universelles des LATs telles que "Président de la République" ou "Pays."

b) Génération de candidats
i. On adapte les techniques d'extraction selon la nature des résultats de la recherche (peut-être que si l'on utilise des ressources 'orientées-titre', le titre sera un candidat)
ii. Affinage du candidat par repérage d'entités nommées, de tf-idf (mesure statistique pour calculer l'importance d'un terme dans un document par rapport à l'ensemble des documents du corpus), triplet rdf..
iii. Si on ne trouve aucun candidat, 0% de chances de trouver le résultat, on fait l'impasse et on saute à la question suivante.
iv. Favorise le rappel sur la précision.
v. Plusieurs centaines de candidats à ce niveau.

4. Soft Filtering
   a) Utilisation d'algorithmes légers pour filtrer les candidats.
   b) Score de « soft filtering » et calcul du seuil en-dessous duquel le candidat n'est pas considéré fiable.
   c) Les candidats qui franchissent le seuil sont transférés vers le calcul hypothèse-preuve.
   d) Utilisation de l'apprentissage automatique (régression logistique) pour le score
   e) Max(100 candidats, paramétrable)
5. Score hypothèse-preuve
   a) Processus d'évaluation rigoureux
      i. Ajout de preuves supplémentaires pour chaque candidat ou hypothèse
      ii. Application de notation profonde (réseaux de neurones) pour évaluer ces preuves supplémentaires

      Récupération des preuves
      i. Technique de la recherche de passage : le candidat est ajouté à la requête de recherche primaire de la question
      ii. Récupère les passages avec le candidat dans le contexte de la question
      iii. A nouveau évaluation des preuves
   b) Notation
      i. Détermine la certitude que les preuves soutiennent le candidat
      ii. Plusieurs annotateurs
      iii. Watson :
         A. possède plus de 50 annotateurs
         B. degré de correspondance entre la structure prédicat argument du passage, et la question
         C. geospatial : présence ou absence de relation spatiale
         D. relations temporelles : incohérences entre les deux entités
         E. classification taxonomique
         F. analyse lexicale et sémantique
         G. corrélation des termes
         H. popularité...
         I. calcul de l'idf du terme
         J. sequence-matching algorithm (taille de la plus grande chaîne de caractères en commun)
         K. représentation en graphes
      iv. Chacune de ces méthodes est indépendante et contribue à améliorer les scores.
      v. Création d'un profil de preuve : création d'une dimension à partir de chaque preuve.
6. Regroupement et classement des candidats

   a) Combinaison de précision (affinage) et de rappel (des centaines de candidats)
   b) Evaluation des hypothèses sur des centaines de milliers de scores.
   c) Estimation du taux de confiance
7. Regroupement de la réponse
   a) Plusieurs réponses pour une question : concept identique mais représentation textuelle de celui-ci différente
   b) Merging : permet d'éviter de trier des réponses qui ont le même sens
   c) Technique du boosting de confiance pour des candidats similaires
   d) Algorithmes de résolutions de co-référence
   e) Combinaison des scores
8. Classement et estimation de la confiance
   a) Approche apprentissage supervisé basé sur des scores connus
   b) Classement et confiance sont séparés
   c) Modèles intermédiaires et types de domaines pour regrouper les ensembles de scores
   d) Techniques de hiérarchisation et généralisation empilée
   e) Méta-apprenant
   f) Amélioration itérative du modèle
   g) Répartition en classe d'entrées (input) pour être le plus efficace possible (type de questions)
   h) Techniques d'apprentissage(régression logistique) pondérées par le taux de confiance
9. Vitesse et échelonnabilité
   a) Apache UIMA pour compresser la durée de traitement et répartir les tâches de façon optimale.
   b) Utilisation du Framework Hadoop pour la mise en grappe de serveurs de calcul

Dans le système DeepQA, la précision signifie le nombre de questions correctement répondues par Watson. Le rappel signifie le nombre de questions auxquelles Watson a décidé de répondre.

La figure 2.4 montre les performances de Watson (courbes) pendant les trois années de développement. Les points gris représentent des joueurs de *Jeopardy !* tandis que les points noirs indiquent que ce sont des joueurs spéciaux ou des champions récurrents.
En 2007 (baseline), pendant les premiers mois de développement, la précision de Watson était de moins de 50% pour une seule question buzzée. Cette précision chutait pour avoisiner les 10% dès que Watson répondait à la moitié des questions. A titre de comparaison, la plus mauvaise performance humaine avec le même rappel de 50% était de plus de 70%.
Après plus de trois ans d'échecs et d'expériences, Watson atteignait le score de 80/80. La meilleure performance du champion de l'époque était de 88/80. Watson avait

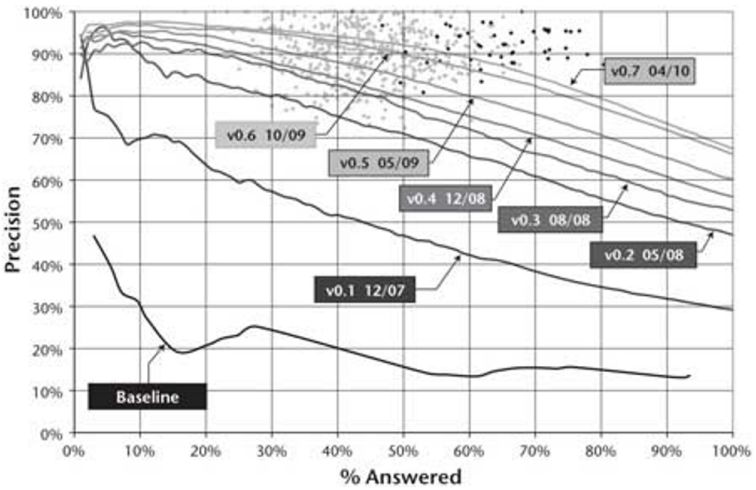


FIGURE 2.4 – tableau de la progression de DeepQA de 2007 à 2010

dépassé le niveau d'un simple joueur et rivalisait donc en termes de performance avec les meilleurs.

Les résultats sont excellents en précision, rappel, mais également en performance car, si en 2007 Watson mettait près de deux heures à répondre à une question, en 2010 il y parvient en moins de cinq secondes. Il est prêt pour le jeu.
En 2011, Watson gagne *Jeopardy !* face au champion en titre, Ken Jennings. Il devient alors mondialement connu.
Cependant, il faut ressituer ce projet dans son contexte : Watson a été conçu pour gagner ce jeu. Les développeurs avaient programmé Watson de telle façon qu'il jouât de façon 'stratégique' en choisissant les indices où il possédât le plus de chances à la fois de répondre correctement et de gagner de l'argent, ou en faisant en sorte que ses rivaux en gagnassent le moins possible ou dussent répondre aux questions les plus susceptibles de leur faire commettre des erreurs. D'autre part, pour parvenir à ce niveau de compétitivité, plusieurs super-calculateurs ont dû être mis en grappe de serveur, et des dizaines de processeurs ont travaillé simultanément, afin d'obtenir la puissance de calcul nécessaire pour les performances requises.

2.1.4 ...et après ?

Après le succès médiatique international de Watson à *Jeopardy !*, l'équipe de développement a évalué la solution DeepQA dans une des campagnes du TREC. Les scores étaient de 51% de précision. Un comparatif des meilleurs résultats mondiaux avec TREC entre 2007 et 2016 est disponible (tableau 2.1).

Les résultats restaient encourageants, montrant la robustesse de leur solution et sa capacité d'adaptation à un autre ensemble de données.
L'objectif dès lors est de déployer les différentes technologies utilisées pendant le projet sous forme de services ou d'applications à des fins commerciales.

TABLE 2.1 – Algorithm - Raw Version of TREC QA & Reference & MAP

Punyakanok (2004)	Wang et al. (2007)	0.419
Cui (2005)	Wang et al. (2007)	0.427
Wang (2007)	Wang et al. (2007)	0.603
H&S (2010)	Heilman and Smith (2010)	0.609
W&M (2010)	Wang and Manning (2010)	0.595
Yao (2013)	Yao et al. (2013)	0.631
S&M (2013)	Severyn and Moschitti (2013)	0.678
Shnarch (2013) - Backward	Shnarch (2013)	0.686
Yih (2013) - LCLR	Yih et al. (2013)	0.709
Yu (2014) - TRAIN-ALL bigram+count	Yu et al. (2014)	0.711
W&N (2015) - Three-Layer BLSTM+BM25	Wang and Nyberg (2015)	0.713
Feng (2015) - Architecture-II	Tan et al. (2015)	0.711
S&M (2015)	Severyn and Moschitti (2015) 0.746	
Yang (2016) - Attention-Based Neural Matching Model	Yang et al. (2016)	0.750
H&L (2016) - Pairwise Word Interaction Modelling	He and Lin (2016)	0.758
H&L (2015) - Multi-Perspective CNN	He and Lin (2015)	0.762
Rao (2016) - PairwiseRank + Multi-Perspective CNN	Rao et al. (2016)	0.780

source :*https : / /aclweb.org*

En 2014, IBM communique la sortie de sa plate-forme Bluemix sur le nuage contenant entre autres les services Watson. Nous pouvons citer Watson Question Answering (le nom est univoque) Natural Language Classifier (apprentissage automatique) Speech To text et Text to Speech...
Les premiers passages en production de Watson n'ont pas été brillants.
Par exemple, il a fallu attendre 2016 et l'explosion de la technologie des agents conversationnels et des classifications par réseaux de neurones à convolution pour que le service Conversation (anciennement Question-Answering puis Dialog, renommé en 2016) montrât qu'il était compétitif..

Aujourd'hui, Watson est très utilisé dans le domaine médical. En effet, sa capacité à analyser des millions de données en quelques secondes font de lui un assistant performant, notamment dans le secteur de l'oncologie (détection automatique des tumeurs les CNN (réseaux à convolution) était très efficaces pour la reconnaissance automatique d'images.). Grâce à ses performances, il est utilisé pour les analyses de la fraude bancaire et fiscale, et aussi dans le secteur juridique.

2.2 Quelques outils Watson

Dans cette section, nous décrivons en détails quatre services et logiciels de Watson que nous avons utilisés pour ce mémoire. Les deux premiers, faisant partie de l'ensemble Watson Explorer, sont réservés aux entreprises et à l'industrie. Ce sont ceux spécialisés dans la fouille de textes.
Le troisième est un service Bluemix servant à classifier des textes par apprentissage automatique. Le dernier est également un service Bluemix servant à créer un *chatbot*.

2.2.1 Watson Content Analytics

WEXCA (Abrégé de Watson Explorer Content Analytics) ne fait pas partie de Bluemix. C'est un logiciel en distribution privée japonais appelé à l'origine TAKMI (Text Analysis for Knowledge MIning) et renommé WEXCA en 2011 lors de la démonstration de Watson à *Jeopardy !* Comme son nom l'indique il s'agit d'un outil

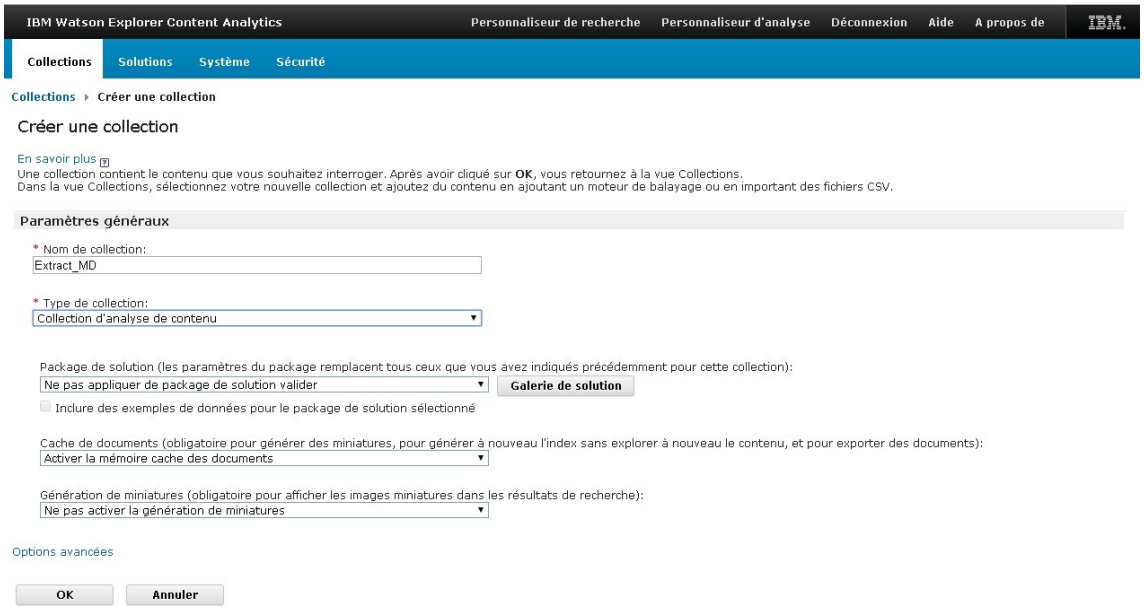


FIGURE 2.5 – Création d'une collection dans Watson Content Analytics

créé pour la fouille de contenu.
Il donne la possibilité de créer un corpus, appelé collection dans le monde d'IBM et de choisir la langue de pré-traitement, bien qu'il ait également un détecteur automatique de langue.
Il est composé de trois outils, l'explorateur, l'indexeur et l'analyseur (figure 2.6).
L'**explorateur** (ou crawler en anglais) sert à rassembler les documents à analyser. Il permet d'explorer le système local, mais aussi des réseaux sociaux, des journaux.. C'est lui qui crée la collection (figure 2.5). A ce niveau, il est possible d'exporter la collection telle quelle.
L'**indexeur** sert à appliquer les facettes (voir section 2.1.1) préalablement créées ou générées automatiquement à la collection. Nous pouvons également choisir quelles facettes appliquer et lesquelles ignorer. A ce niveau, il est possible d'exporter la collection 'annotée' en facettes sous forme d'XMI, un format XML (langage de balisage extensible) spécifique à UIMA.
Enfin, l'**analyseur** est l'outil permettant d'effectuer les requêtes, d'observer le comportement des facettes dans le temps (écart et tendance que nous avons mentionnés précédemment), ou par rapport à d'autres facettes. A ce niveau, il est uniquement possible d'exporter les résultats de ces requêtes, sous forme d'XML ou de JSON.

L'analyse de contenu

La fenêtre graphique d'analyse de contenu permet de d'observer les documents, pour vérifier que l'exploration est l'indexation se sont bien déroulées. C'est dans cet onglet que nous pouvons regarder si nous avons rencontré des problèmes d'encodage, de doublons...
La fenêtre propose une section "**facettes**" où nous pouvons observer tout ce que le système a trouvé comme représentations linguistiques de ces facettes (figure 2.7)
Pour anticiper la section suivante, la figure 2.8 montre deux facettes différentes :

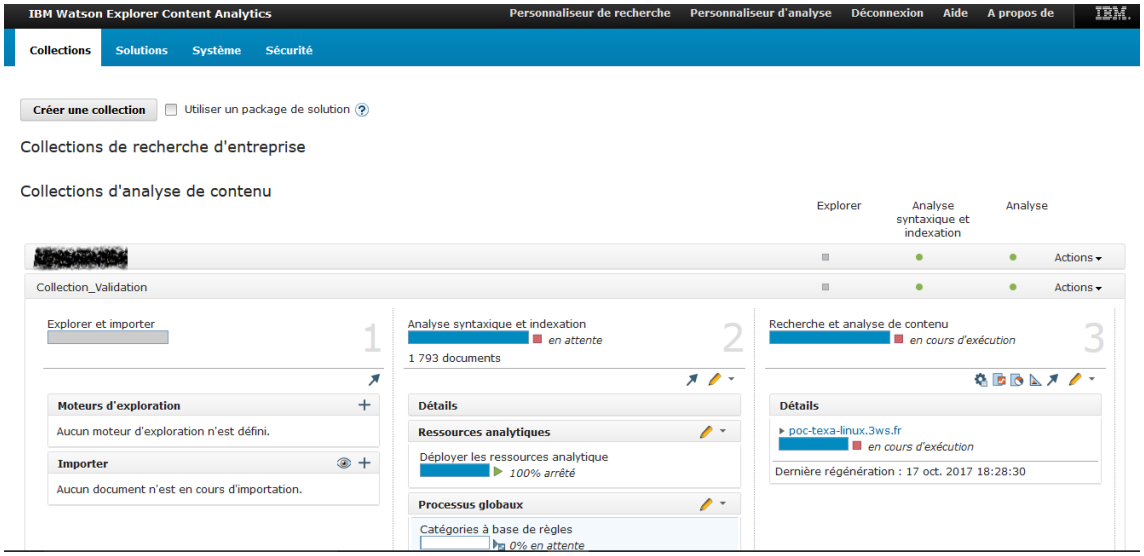


FIGURE 2.6 – Console d'administration Watson

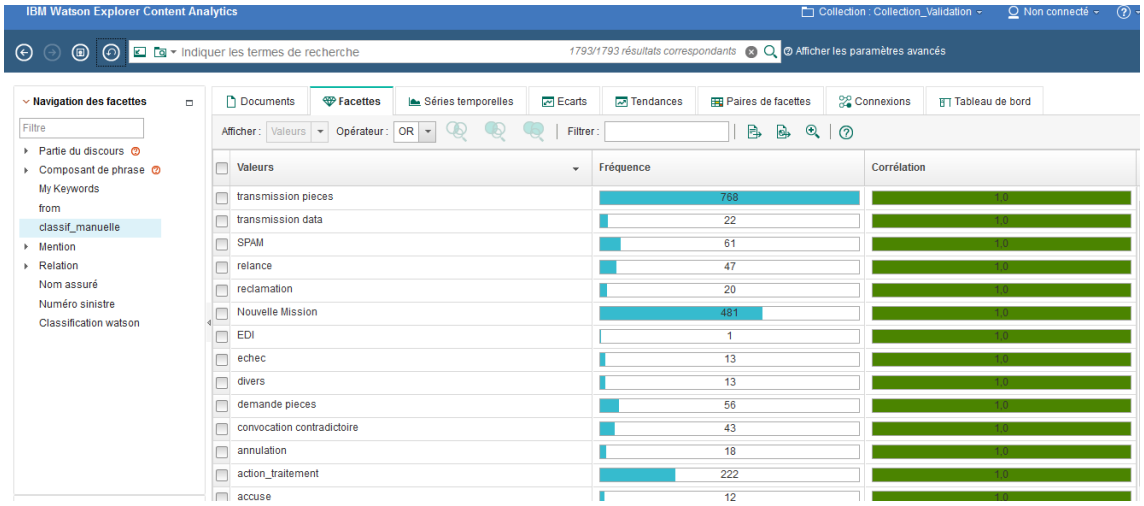


FIGURE 2.7 – nombre de documents classifiés par valeur de facettes en classification manuelle

notre classification manuelle (corpus de référence) et la classification de Watson à partir de règles : (voir section 2.2.2). Sur cette figure, nous pouvons apercevoir une valeur de facette appelée *'Nouvelle Mission'*.Elle a été sélectionnée à partir de la classification de Watson. Grâce à l'outil de requêtes, il est possible savoir comment chaque document classé comme *'Nouvelle Mission'*, l'était dans la classification manuelle. En d'autres termes, nous évaluons ainsi la précision de l'annotateur Watson sur une thématique particulière. Dans la figure nous pouvons observer en bleu les occurrences de chaque classification manuelle selon les *Nouvelle Mission* de Watson. Nous observons que la plupart des occurrences se trouvent également dans *Nouvelle Mission* de la classification manuelle. Malgré un peu de bruit avec quelques occurrences dans la classe *Transmission pièces*, l'évaluation est très positive.
A droite, les données coloriées en vert montre la corrélation entre deux classifications. Cette donnée corrobore notre hypothèse car nous pouvons observer comment

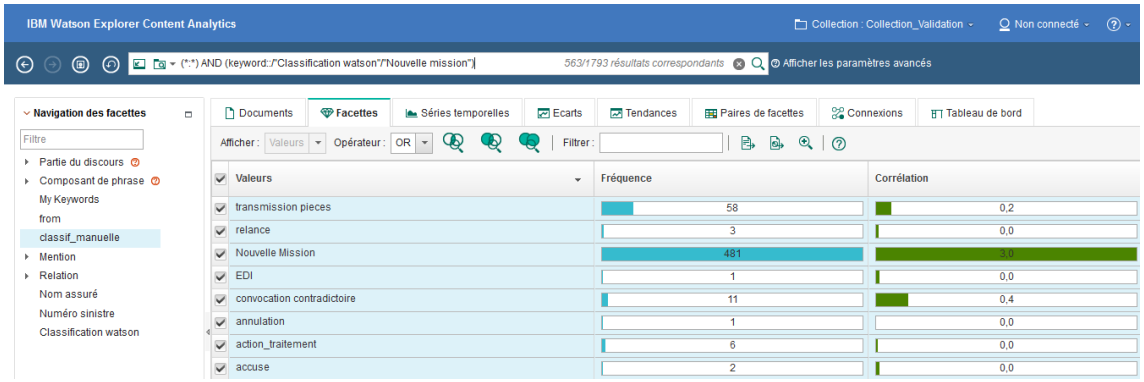


FIGURE 2.8 – Classification des Nouvelles Missions selon Watson et vérification selon la classification manuelle (précision)

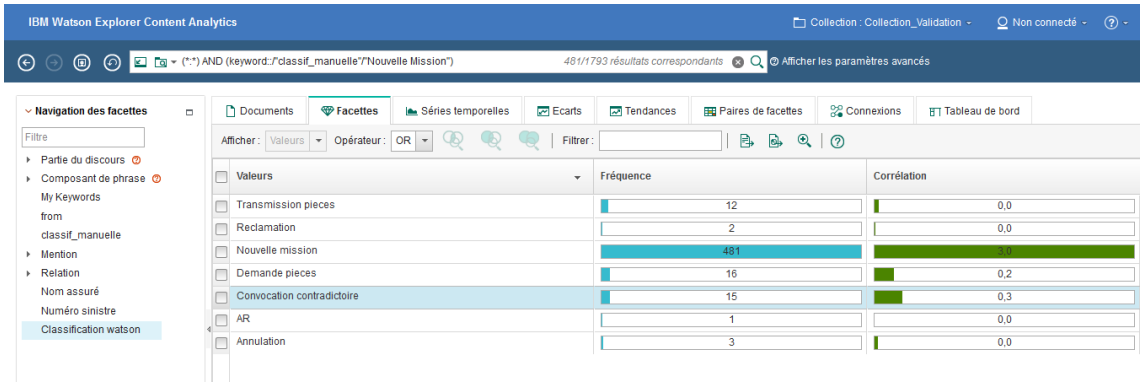


FIGURE 2.9 – Classification manuelle des Nouvelles Missions et vérification de ce qu'a trouvé Watson (rappel)

Nouvelle Mission pour Watson et pour l'annotateur humain sont fortement corrélées.

Le rappel se fait, à l'inverse, en sélectionnant les nouvelles missions dans la classification manuelle et en observant comme les a classées Watson (figure 2.9)

Que ce soit pour le rappel ou pour la précision, nous avons utilisé une requête de la forme :
Sélectionner toutes les valeurs de la classification Y qui ont pour valeur Z dans la classification X.
Voici fonctionnalités de création de règles linguistiques dans Watson.

2.2.2 Watson Content Analytics Studio

L'outil *Watson Content Analytics Studio*, abrégé en Studio, permet de créer des annotateurs personnalisés, qui viennent s'ajouter en dernier dans la chaîne de traitement UIMA. Ces annotateurs étant construits par l'annotateur humain, ils sont les ultimes à être pris en compte, donc ils sont les plus importants.
Studio est un logiciel écrit en java, l'interface graphique est d'ailleurs basée sur Eclipse. Il permet à l'utilisateur, plutôt que de créer toutes les règles en code, de les construire de façon intuitive avec un éditeur graphique.

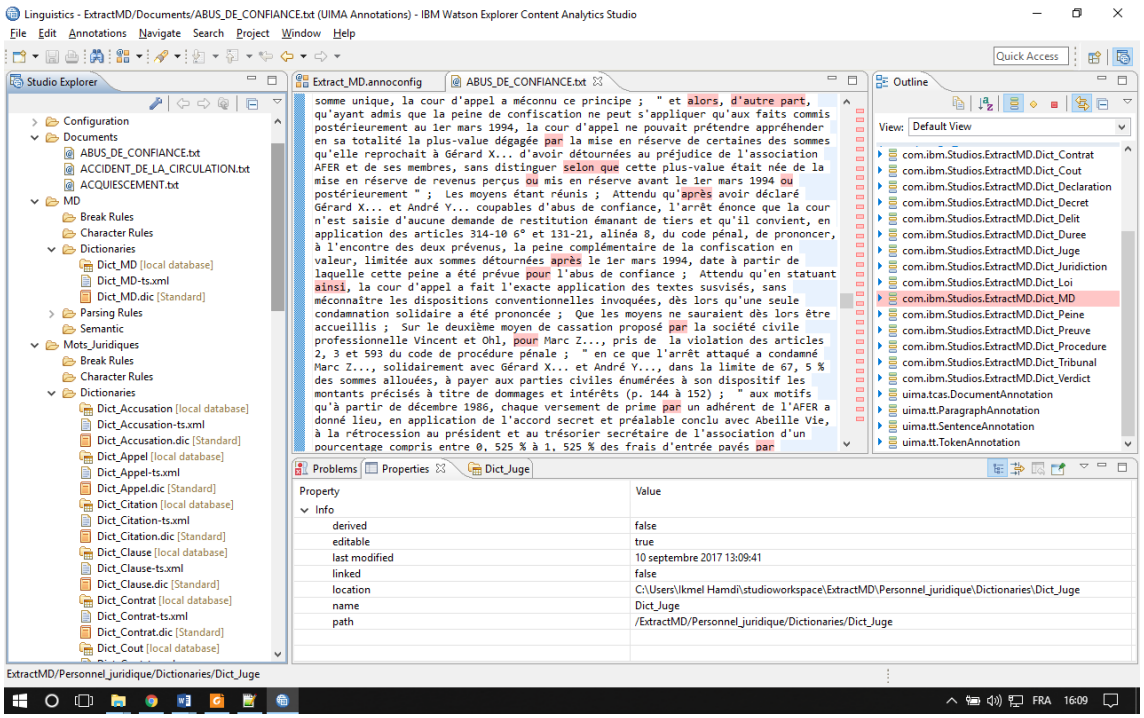


FIGURE 2.10 – création d'une règle dans Studio

Après avoir choisi la langue de l'annotateur, nous importons un ou plusieurs documents dans l'interface et nous lançons l'analyse morphologique par défaut.
Nous pouvons ainsi *"‘ GLISSER et DEPOSER "‘* un syntagme de ce document dans un onglet, et le manipuler selon les besoins (lemmatisation, segmentation, suppression d'éléments, utilisation d'expressions régulières...) afin de créer une règle, elle-même servant à créer une facette.
Studio permet d'observer, à partir d'une règle nouvellement créée, quels autres syntagmes sont reconnus par celle-ci (figure 2.10). Ainsi, nous pouvons examiner lesquels ont été oubliés, et créer une nouvelle règle, pour englober les autres représentations textuelles de la même facette. De même, nous pouvons identifier les 'faux positifs' et les utiliser comme exemples de syntagmes ne devant pas déclencher la règle en question.

Enfin, nous pouvons en tirer un enseignement : le fonctionnement de Studio nous offre le choix entre deux manières de travailler. La technique la plus évidente est celle de l'approche empirique. Nous lisons le document qui sert d'atelier de travail, nous en extrayons des mots qui semblent significatifs, nous en faisons des dictionnaires, puis nous combinons ces dictionnaires pour créer des règles, nous combinons les règles entre elles et ainsi de suite. Cela fournit un nombre réduit de facettes entièrement construites par la connaissance empirique du texte.
Cependant, ce n'est pas la seule possibilité. Il existe le cas de figure où les facettes à alimenter sont connues et décidées à l'avance. De plus, nous possédons déjà des exemples linguistiques de celles-ci. Ainsi, en décomposant ces exemples, nous parvenons à créer des sous-facettes, puis le niveau atomique, des dictionnaires (un seul mot en général) pour analyser l'intégralité du document de façon descendante et lorsque qu'il le faut corriger ou augmenter la portée d'une règle ou encore la rendre

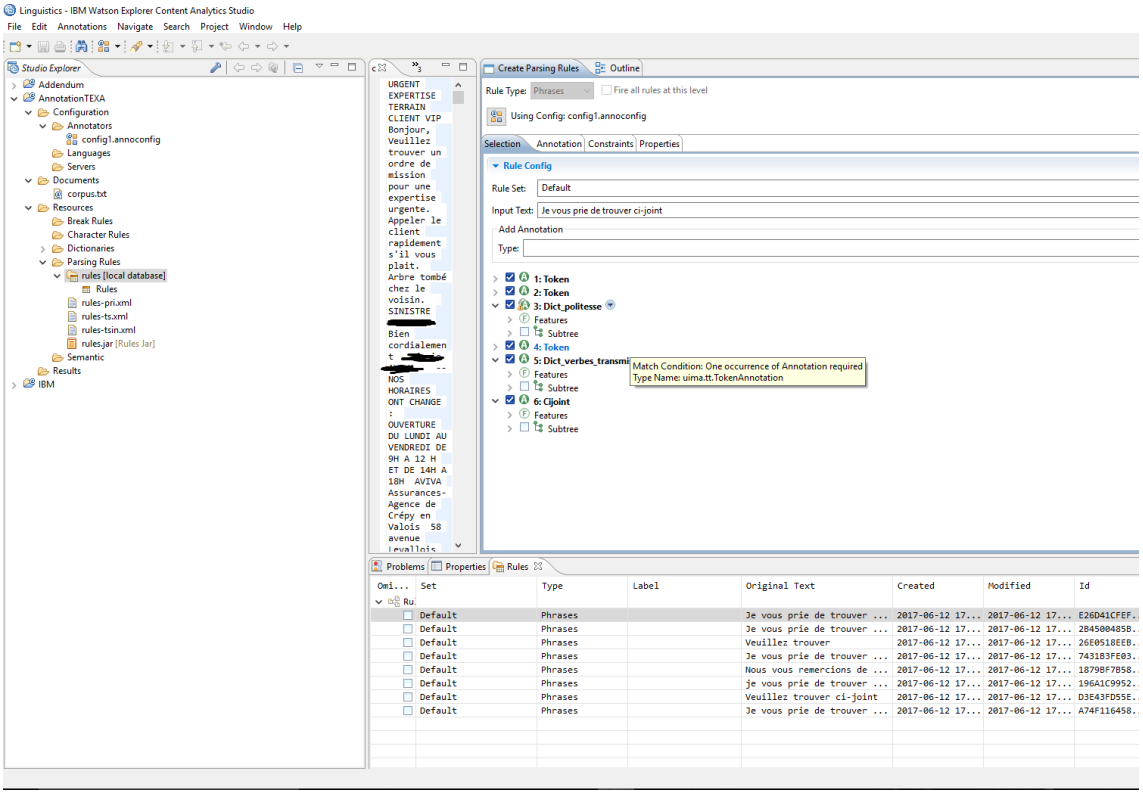


FIGURE 2.11 – Composition d'une règle

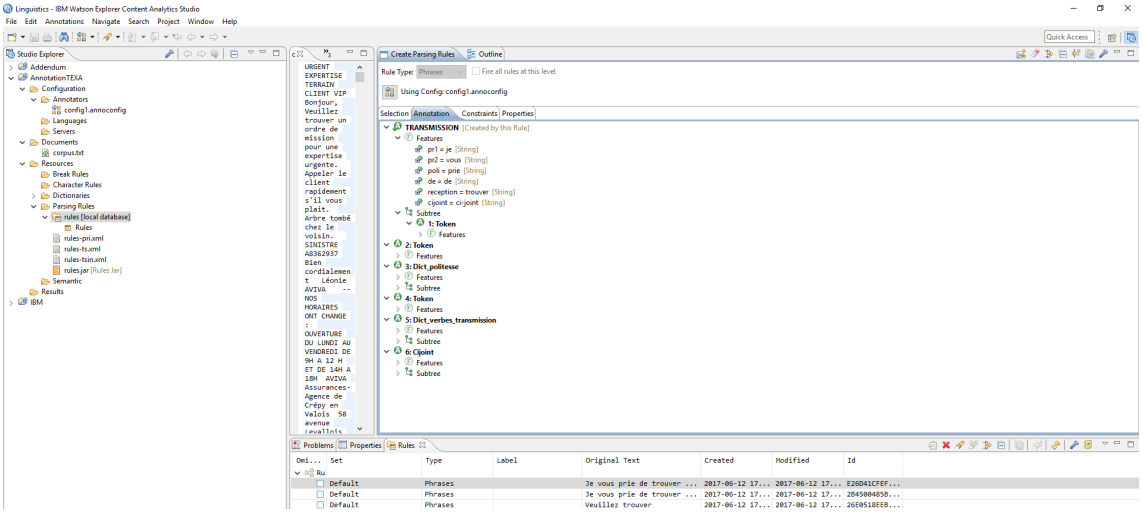


FIGURE 2.12 – Exemple de facette

moins permissive.
La seconde approche présente l'avantage d'être plus rapide dans la mesure où nous possédons une liste de ce qui est attendu. Par contre, la première approche, puisque nous partons de rien, ne se limite pas à des exemples déjà fournis, donc les variations linguistiques voire les facettes que nous pouvons découvrir sont *a priori* illimitées. Les figures 2.11 et 2.12 montrent une facette et les règles qui la composent.
Après avoir fini de créer toutes nos règles et établi toutes nos facettes **terminales**, Studio se connecte à Watson Content Analytics, et y importe les facettes en prenant

soin de préciser quelles facettes seront utilisées.
Dans le cadre de notre étude, notre seule et unique facette se nomme *Classification Watson* et chaque valeur de cette facette est en réalité chaque sous-facette créée dans Studio et englobée dans la facette générale 'Classification Watson' en propageant uniquement son nom *(nouvelle mission, transmission données, réclamation, annulation mission...)* comme attribut de 'Classification Watson'.
Cependant, malgré l'aspect intuitif et ergonomique de l'outil, ainsi que son aspect pratique car facile à intégrer à Content Analytics pour la fouille textuelle, cela reste un travail très long, car pour que le résultat soit satisfaisant et le projet Studio réutilisable, il faut réfléchir à quelles facettes construire, pour chacune d'entre elles, nous il faut plusieurs règles, chaque règle a besoin de plusieurs dictionnaires. Ainsi, il faut construire des dizaines de sous-facettes et de règles, et des centaines de dictionnaires. A titre d'exemple, pour notre système Question-Réponse, la création des règles a pris plus d'un mois.

2.2.3 Watson NLC

Nous abordons la partie apprentissage automatique dans Watson. Il existe aujourd'hui plusieurs outils dans Watson : **NLU (natural language understanding)**, le dernier en date, servant surtout pour la reconnaissance d'entités nommées (noms propres, noms de lieux, noms d'organisations...), ou de chaînes de caractère spéciales (codes, numéros de dossier, dates..), **Watson Knowledge Studio** qui est à NLU ce que Studio est à Content Analytics, et **Retrieve and Rank**, servant à donner des scores à des réponses selon les questions fournies, devenu aujourd'hui obsolète.
Celui qui nous intéresse s'appelle **Natural Language Classifier (NLC)**. C'est un service Bluemix, directement issu du projet DeepQA et qui se révèle plutôt performant, en tout cas pour les usages que nous en avons faits. C'est une implémentation de **réseaux de neurones à convolutions** combinés à des **règles linguistiques** propres au traitement automatique des langues.
Il prend en entrée un groupe de documents, annoté en classes. Chaque document peut appartenir à une ou plusieurs classes. Chaque classe doit posséder au moins dix instances. Sans cela, la classification échoue.
A partir de ces documents. NLC entraîne un classifieur.
Ensuite, pour chaque document donné en entrée du classifieur, il renvoie un JSON (format de données textuelles dérivé de la notation des objets du langage JavaScript.) contenant toutes les classes, dans l'ordre décroissant du taux de confiance avec lequel NLC estime que le document appartient à la classe.
Par exemple, soit le vecteur de document X contenant des mots de x_0 à x_n, donné en entrée au classifieur, sachant qu'il existe trois classes différentes notées A B et C, le classifieur donne la réponse suivante : **{**A**= 78%;** B **= 7%;** C**= 2%}** Le total n'est pas forcément égal à 1 puisque les taux de confiance de chaque classe sont indépendants des autres.
Tout comme pouvait le faire DeepQA, nous décidons d'imposer un seuil à NLC en dessous duquel même sa meilleure réponse avec le taux de confiance le plus élevé n'est pas prise en compte. Par exemple, si nous fixons le seuil à 80%, le JSON renvoyé est vide et à la place il renvoie la réponse par défaut, comme 0.

2.2.4 Watson Conversation

Enfin, le dernier outil exploité est l'agent conversationnel(*chatbot*), Watson Conversation. Nous y programmons notre agent en interface graphique. Pour ce faire, nous créons trois objets : les **entités**, les **intentions** et l'**arbre de dialogue** (qui est le seul indispensable).

entités

Les entités (figure 2.13) représentent les **critères** que le robot doit reconnaître afin de diriger le dialogue vers une branche spécifique et ainsi donner la réponse adéquate, ou pour capturer des informations dont le robot doit se servir à un moment donné de la conversation. Par exemple, pour un robot spécialisé dans les problèmes de *déplacement quotidien travail-domicile*, nous créons l'entité *'Transports'* que nous instancions avec la valeur *'train'*. Cependant, dans la mesure où nous ne savons pas quelle représentation linguistique de *'train'* l'utilisateur risque d'employer, nous devons créer des synonymes appartenant à différents registres : "voie ferrée","VF","TGV","TER","ferroviaire"...
D'autre part, il arrive que l'utilisateur, dans la précipitation, n'écrive pas bien le critère. Celui-ci devient un faux-négatif et ne serait pas détecté par le robot comme un critère. Par exemple *'trani'*.
Pour prévenir ce problème, le service dispose de la **correspondance approximative** (fuzzy matching en anglais). Cette option utilise la distance de Levenshtein, une distance mathématique permettant dans ce cas-ci de calculer la proximité orthographique entre deux mots.
A noter qu'il existe des entités systémiques : par exemple, les nombres, écrits en chiffres ou en lettres, les dates (de l'heure à l'année), les devises sont reconnues par défaut dans l'outil. Et ce en plusieurs langues. Ces entités peuvent être activées.
Dans le *chatbot* Watson, les entités sont représentées par **'@'**.

Intentions

Les intentions (figure 2.14) représentent les types de phrases, de questions, ou d'ordres que donnent les utilisateurs au robot. Par exemple, pour un robot spécialisé dans les questions administratives, il existe une intention appelée *'Demande de remboursement',*. Dans cette intention nous stockons toutes les représentations linguistiques possibles de cette intention : "Serais-je remboursé si je dois dîner dehors ?" ;"Quel type de remboursement pour mes déplacements quotidiens ?" ;" Je voudrais me faire rembourser ma dernière nuit d'hôtel" Ces exemples textuels sont aussi appelés **instances**.
Contrairement aux entités, les intentions doivent être entraînées. En effet, les valeurs d'une intention étant pour la plupart des phrases ou des bouts de phrases, le robot analyse chaque mot de la phrase pour tenter de comprendre ce que signifie cette intention, et pour une nouvelle phrase donnée en entrée par un autre utilisateur, déterminer si elle appartient à cette intention. Le robot apprend les intentions au moyen d'un service que nous avons mentionné précédemment : NLC. De plus, lorsque nous entraînons le robot, nous pouvons le corriger : si jamais la phrase en entrée ne correspond pas à l'intention comprise par le robot, on lui signale laquelle est la bonne, si tant est qu'il existe une intention correspondant à cette phrase dans

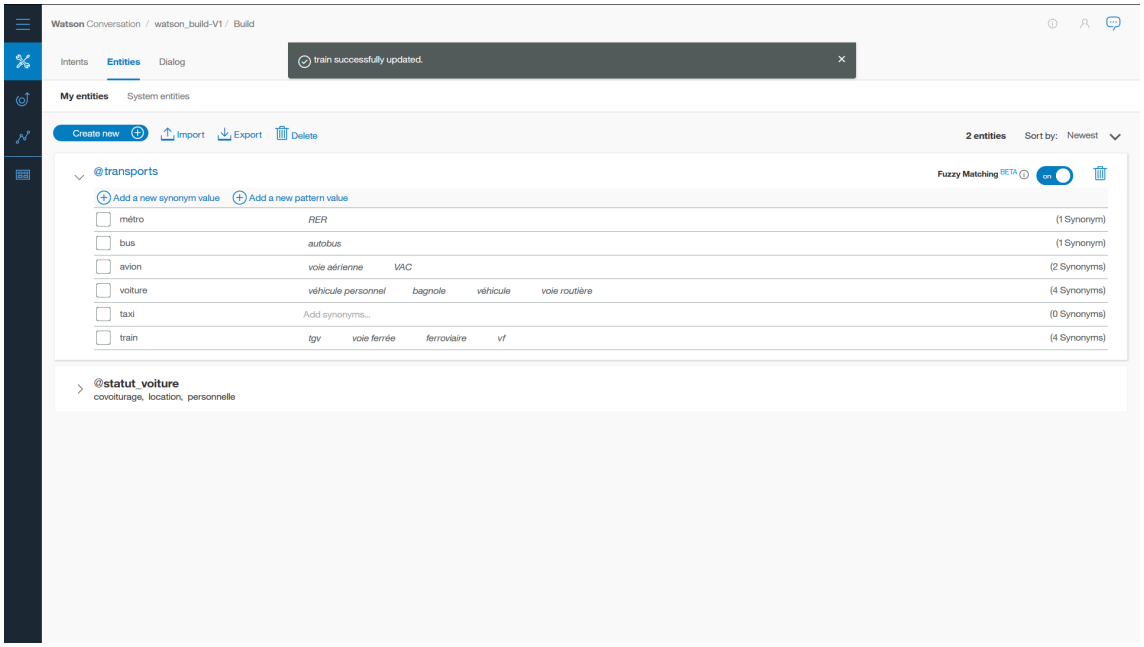


FIGURE 2.13 – Création des entités

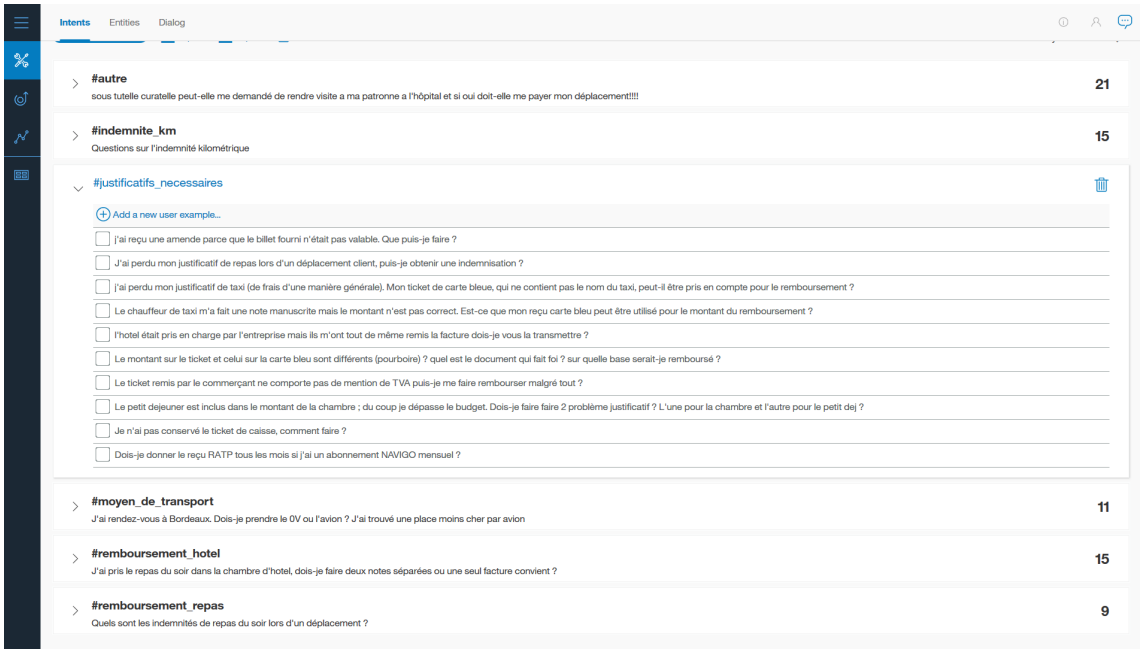


FIGURE 2.14 – On crée les intentions

l'architecture.
Alors que l'on peut parler de détection pour les entités car il s'agit de règles basiques (correspondance exacte ou approximative), on peut parler d'interprétation pour les intentions. Dans le *chatbot* Watson, les intentions sont représentées par **'#'**

dialogue

Le dialogue constitue le cœur du *chatbot*. Nous pourrions nous passer des intentions et des entités pour ne manipuler que l'entrée "en dur" de l'utilisateur. Par exemple un *chatbot* servant de calculatrice, n'a aucun mal à analyser des entrées simples *(4+3, 5.1/78...)*. Cependant, cette manipulation de l'entrée brute se faisant à base de règles, donc d'expressions régulières, elle demeure limitée (automate à états finis). C'est pourquoi les intentions, et surtout les entités sont nécessaires dès lors que nous souhaitons mettre en place une solution cognitive.

Le dialogue est constitué de branches, représentées dans l'outil Watson comme des boîtes (voir figure 2.15). Chaque branche ne peut venir que d'un et d'un seul parent, mais peut avoir plusieurs des filles, et plusieurs sœurs. Chaque branche peut posséder un nom qui l'identifie (utile pour déboguer lorsque notre dialogue contient des centaines de branches). Elle peut également posséder une condition d'activation initiale. Par expérience, même lorsqu'aucune condition n'est requise, il est préférable de lui mettre en condition **" if TRUE "**, qui est une condition vraie en toute circonstance, plutôt que rien, cela évite des effets de bord. Cette condition initiale est suivie d'une réponse. La réponse peut être vide (le dialogue ne répond rien à l'utilisateur). Elle peut être également accompagnée d'un contexte. Ce contexte est très important car c'est ce qui permet de propager les informations de l'utilisateur d'une branche à l'autre. En effet, le *chatbot* dans Watson n'est pas une machine à états complets.Il ne conserve dans sa mémoire que les informations contenues dans l'entrée actuelle de l'utilisateur. Autrement dit, soit un *chatbot* météo, si l'utilisateur lui demande quel temps il fait aujourd'hui à Marseille, il sera incapable de répondre à la question suivante, **"et demain ?"**. Le contexte permet, en bricolant une règle, de résoudre ce problème en stockant dans des variables de contexte les informations nécessaires à la poursuite du dialogue. Dans le chapitre sur les expériences, nous expliquons la raison pour laquelle ces variables de contexte ont été indispensables.
Voici un exemple de branche de dialogue de livraison de pizzas :

INPUT :
« C'est possible d'enlever les champignons et de mettre des olives à la place ? »

ANALYSE DE WATSON :
? 'c'est possible d'enlever' -> #enlever

'mettre des ? à la place' -> #remplacer

'champignons -> @ingredient :champignon

'olives' -> @ingredient :olive.

Dans son analyse, il stocke dans un tableau toutes les intentions qu'il a repérées et renvoie celle dont le taux de confiance est le plus élevé. La probabilité qu'un utilisateur utilise exactement la même intention que celles pré-construites par le développeur est presque nulle. C'est pour cela que Watson va calculer le taux de proximité entre les intentions qu'il connaît et l'entrée de l'utilisateur. S'il se trompe, il suffit de lui indiquer quelle est la bonne intention, et il l'apprend pour la prochaine occurrence.

Exemple :
« **C'est possible d'avoir les champignons en moins ?** »

'en moins' = #prix reduction.

'champignons' = @ingredient :champignon

Dans ce cas-là, Watson considère que l'intention la plus probable est que l'utilisateur désire savoir si les pizzas sont moins chères, bien qu'il trouve aussi l'entité **@ingrédient**. Il faut donc lui indiquer (manuellement) quelle est la bonne intention, à savoir **#enlever**.
Comment se servir alors de l'analyse de Watson pour générer une réponse satisfaisante ? Il suffit de créer des branches de dialogue qui se déclenchent selon des conditions univoques.
IF #ENLEVER AND @INGREDIENT AND #REMPLACER.
Le traitement devient aisé, car les cas de déclenchement ne sont pas ambigus.
On peut remarquer qu'il n'est pas demandé dans la condition l'apparition d'un second ingrédient après l'intention de **#remplacer.** Cela est dû au fait que cette syntaxe (# et @) désigne un booléen.
(if exists in input) Donc cela ne sert à rien de répéter
AND @INGREDIENT car le AND n'est pas un 'et en plus'.
La question qui se pose est comment faire pour traiter les cas d'apparition de plusieurs occurrences de la même entité. Nous y répondons plus bas.

Après une réponse du robot, trois cas de figure existent :

1. *Le chatbot attend un input de l'utilisateur*
   -> « Quelle pizza voulez-vous ? » -> « ce sera tout ? » -> « votre numéro de tel svp »
2. *Le chatbot passe directement à un autre nœud sans attendre d'input :*
   « Votre commande est validée »
   |
   « Cela vous fait un total de 15 euros »
   |
   « Votre pizza sera livrée d'ici 30 minutes, vous pourrez régler par carte ou en espèces auprès du livreur »
   Dans ce sous-dialogue, le *chatbot* n'attend pas d'input, il traite les données qu'il a enregistrées au fur et à mesure de la conversation (variables de contexte) et conclut son dialogue.

3. *Le chatbot passe directement à un autre noeud PUIS attend l'input de l'utilisateur.*
   Ce cas-là est très utilisé lors des boucles : si l'utilisateur entre une information non compréhensible, le *chatbot* lui demande de répéter, ou le transfère vers un nœud plus pertinent pour le traitement de la requête de l'utilisateur, qui doit tout de même la répéter.

Un problème auquel nous avons fait face lors de notre travail a été la gestion des niveaux de difficulté d'une entrée de l'utilisateur.

TABLE 2.2 – Dix niveaux de difficulté

0 : commande fausse, mauvaise syntaxe	ERREUR
1 : un seul objet de même nature sans option	(2 margarita)
2 : plusieurs objets de même nature sans option	(2 margarita et 1 regina)
3 : un seul objet (avec option OU avec un autre objet de nature différente)	(1 margarita avec supplément fromage OU 1 margarita avec 1 coca)
4 : [plusieurs objets de même nature (avec même option OU avec un autre objet de nature différente)] OU [deux objets de nature différente] OU [un objet et plusieurs options]	(2 margarita et 1 regina avec suppléments fromage OU 2 margarita et 1 regina avec 2 cocas) OU (2 margarita avec supplément fromage et sauce piquante OU 2 margarita avec 1 coca et 2 fanta)
5 : deux objets de nature différente avec une seule option	2 margarita avec supplément fromage et 2 cocas
6 : plusieurs objets de même nature (avec plusieurs options OU avec plusieurs objets de nature différente)	2 margarita dont une avec supplément fromage et l'autre avec sauce piquante et 1 regina OU 2 margarita et 1 regina avec 1 coca et 2 fanta
7 : un seul objet avec une option et plusieurs autres objets de nature différente OU deux objets de nature différente avec options différentes	2 margarita avec supplément fromage et avec un coca et un fanta OU 2 margarita avec supplément fromage et des poivrons et 1 coca
8 : plusieurs objets de même nature avec options différentes et avec un autre objet de nature différente OU plusieurs objets de nature différente et une seule option	2 margarita dont 1 avec supplément fromage et 1 regina avec sauce piquante et 1 coca OU 2 margarita et 1 regina avec sauce piquante et avec 2 cocas et 1 fanta
9 : plusieurs objets de nature différente avec options différentes	2 margarita avec supplément fromage dont une avec sauce piquante et 1 regina sans champignon et avec 1 coca et 3 fanta, le coca light si possible

Par niveaux de difficulté nous entendons la totalité des opérations qui découlent des combinaisons entre des objets pleins(entités ou intentions) que peut contenir une requête de l'utilisateur, pour un nombre d'objets différents et de quantité donnés. Autrement dit, pour une question posée l'utilisateur **quelle est la difficulté maximale** n **qu'elle peut atteindre et comment en déduire sa complexité maximale** après traitement de toutes les composantes de la difficulté ?
Le tableau (2.2) est un récapitulatif de tous les niveaux de difficulté, pour un *chatbot* de livraison de pizza : En définitive, dans le cadre de notre travail, nous avons décidé de nous concentrer sur des entrées ne dépassant pas le niveau 4. De fait, plus de 95% des entrées ne dépassant pas le niveau 3, nous ne perdons que peu de données.

2.3 Limites de Watson

Finissons ce chapitre par discuter des inconvénients de la solution Watson.

Premièrement, le moteur linguistique derrière **Content Analytics** reste rudimentaire. Par exemple, il n'existe aucune analyse de profondeur, ni syntaxique, encore moins sémantique. Il existe une analyse de sentiments mais avec des résultats moyens. Cela pose un problème important : l'analyse demeure au niveau morphologique. C'est assez paradoxal dans la mesure où pour *Jeopardy !* il existe un niveau d'analyse syntaxique profonde permettant d'exploiter entre autres le système continu verbes-arguments-circonstants.
Deuxièmement, les contraintes techniques sont nombreuses : il faut une machine puissante, et connectée en permanence. Dans **Studio**, le document utilisé pour créer les règles ne doit pas dépasser une certaine taille. Enfin, la configuration des collections dans l'outil demeure longue, et peu intuitive.

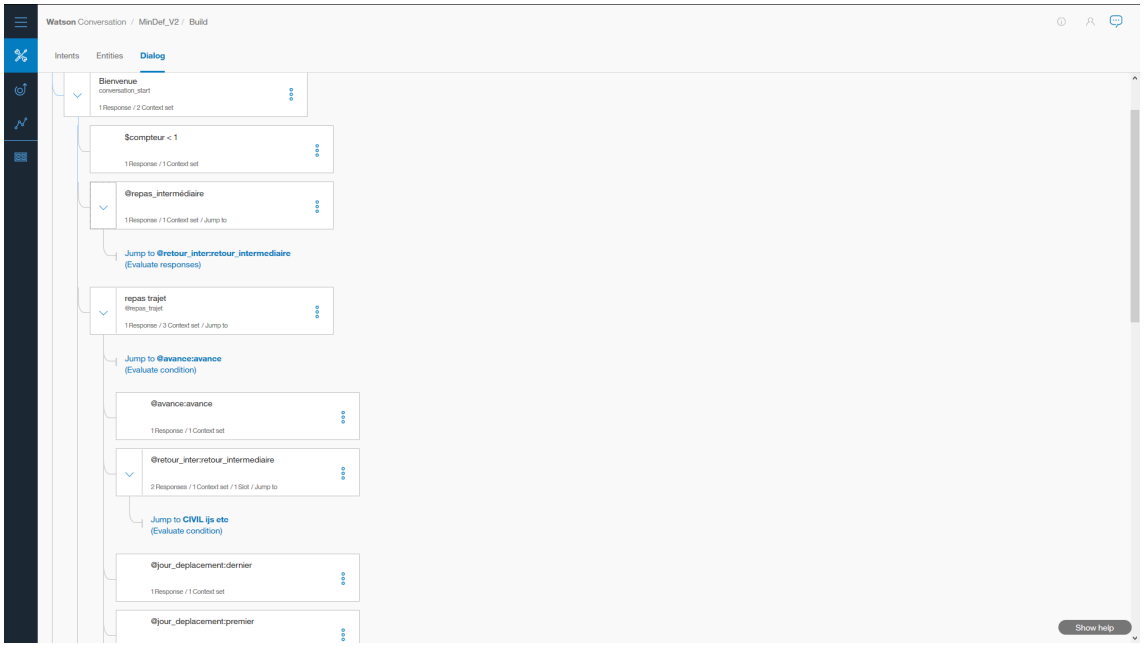


FIGURE 2.15 – Mise en place du dialogue

Troisièmement, **NLC** se présente comme une boîte noire que l'on utilise pour classifier. De ce fait, nous n'avons aucune maîtrise sur l'outil. Etant donné qu'il s'agit de Réseaux de Neurones Artificiels, la configuration du système en fonction des données est très importante. L'opacité du service empêche de modifier certains paramètres pouvant significativement améliorer les résultats de base. Nous verrons dans le chapitre sur les méthodes que l'utilisation d'outils en libre-accès pour l'apprentissage automatique se révèlent plus intéressants dans notre travail.
Pour finir avec **NLC,** la taille de chaque texte en entrée ne doit pas dépasser les 1024 caractères. Autrement dit, nous sommes forcés de nettoyer notre corpus pour obtenir ces 1024 caractères, ou de supprimer arbitrairement tous les caractères après le 1023è.

En dernier lieu, pour Watson Conversation, la sortie récente du produit (un an) explique certains inconvénients. Toutefois, notons quelques problèmes qui ont pu freiner notre travail.

Sur le plan linguistique tout d'abord, les entités, contrairement aux règles et aux facettes de Studio, ne peuvent pas se combiner pour créer des super-entités. Cela ralentit beaucoup la création d'un dialogue dans la mesure où nous sommes contraints de multiplier les entités, et de complexifier les conditions à l'intérieur des branches.
De plus, lorsque nous créons les intentions, il n'est pas possible d'inclure dans les exemples des phrases contenant des entités, sous peine d'une erreur de syntaxe, même pour les entités systémiques.
Ensuite, les entités ne peuvent pas être corrigées, contrairement aux intentions, si Watson se trompe dans leur interprétation. Cela arrive si le **fuzzy matching** n'est pas activé, mais aussi lorsqu'il l'est. Par exemple, si une entité possède des

abréviations, comme c'est le cas dans notre travail pour l'entité transports, *(train : voie ferroviaire, vf...)*, le nombre de faux-positifs de *train* augmente considérablement et finalement le **fuzzy matching** devient inutile car trop bruyant.
Enfin, les entités ne sont considérées par Watson que comme des chaînes de caractères et non comme des mots. De ce fait, aucune analyse morphologique ou lemmatisation n'est effectuée sur elles. Cela nous a contraint à multiplier leurs valeurs et leurs synonymes.

En définitive, ce qu'il faut retenir c'est la performance de Watson, qui possède à lui tout seul près d'une trentaine de services, dont nous nous servons comme des briques utilisables dans des applications ou dans d'autres briques. C'est un outil très puissant pour le Traitement Automatique des Langues mais aussi pour la Recherche d'Information. Malgré tout, il demeure défectueux et incomplet dans plusieurs domaines.
Nous n'avons pas évoqué son prix mais Watson coûte très cher. Dans le cadre d'un projet industriel, le coût pourrait être amorti parce que Watson, se présentant comme un tout-en-un, permet de gagner un temps considérable par rapport au libre-accès qui force à chercher pendant des mois des outils pour remplir toutes les fonctions que Watson propose. Pour un projet scientifique, cet argument ne tient pas et sa pertinence est d'autant moins avérée.

CHAPITRE 3

DESCRIPTION DE LA TÂCHE

Sommaire

3.1 Introduction

Dans ce chapitre, nous nous attardons sur une description de notre chaine de traitement. Nous expliquons quels étaient les objectifs initiaux, les changements imprévus que nous avons dû effectuer, ainsi que différentes briques de notre outil.
Notre système de question-réponse, comme nous l'avons présenté en introduction, se situe dans le cadre de la gestion des problèmes d'indemnisation de personnel militaire en période de stage ou de mission. Nous détaillons aussi dans ce chapitre quels sont ces problèmes.
Enfin, Nous parlons des obstacles et des situations inattendues rencontrés.

3.2 Les objectifs

Le but du système de question-réponse est la création d'un système de question-réponse pour assister employé administratif en charge de répondre aux questions du militaire.
L'employé reformule la question initiale du militaire de façon plus formelle que ce dernier, de façon à ce que le robot puisse l'interpréter (thématiques et critères).

Après être parvenu à analyser la question, le robot évalue ses mots-clés grâce à un dialogue pré-construit. Il détermine alors s'ils sont assez nombreux et corrects

pour effectuer la recherche de la réponse, sinon il interroge directement l'employé militaire pour récupérer les informations manquantes ou erronées.
Lorsque la question analysée est complétée, le robot l'envoie au moteur de recherche. Celui-ci indexe la question et, par un système de pondération des thématiques et des mots-clés, il renvoie au robot la réponse la plus certaine. Le robot génère alors une phrase en langue naturelle répondant à la question de l'employé. La réponse que le robot exploite pour générer sa phrase est un paragraphe de la documentation réglementaire structurée par thématiques, sous-thématiques et critères, eux-mêmes pondérés.

Enfin, le robot interroge l'utilisateur pour savoir si la réponse le satisfait, et en cas de négation, il renvoie la deuxième réponse la mieux classée, et supprime la première des candidats à répondre à la question selon cette combinaison de thématique et de critères. Ainsi, il **apprend** que ce n'est pas la bonne réponse pour ce type de questions.
Nous décrivons maintenant chacune des étapes pour la réalisation de ce système.

3.2.1 Analyse de la question

La tâche plus importante consiste en l'analyse de la question. Nous insistons sur son importance car, même si le *chatbot* et le moteur de recherche sont absents, avec la thématique et les critères correctement détectés, nous pouvons guider l'utilisateur vers les bons paragraphes de la documentation.
Nous avons construit des lexiques, puis des règles pour effectuer cette analyse.
Nous avons relevé une quinzaine de thématiques différentes dans les questions des militaires. Pour ce mémoire, nous avons décidé de n'en conserver que trois, et de spécialiser notre analyse dans la détection de ces trois thématiques : **" retour intermédiaire", "repas-trajet", "principe de remboursement".**

Retour intermédiaire : Cela concerne les questions où le personnel militaire souhaite savoir s'il sera remboursé, pendant un stage où une mission, de ses déplacements les week-ends, lorsqu'il veut rentrer chez lui, à quels taux, et dans quelles conditions.
Exemple : *" **Question :** Bonjour Quel est le **mode de remboursement** pour les élèves de Salon qui sont en stage sur une autre base et qui lors des **retours intermédiaires** retournent à leur domicile. merci d'avance. **Réponse :** Bonjour Votre demande concerne les **retours intermédiaire** pour un stage.",1* Dans cet exemple, notre système de règles détecte deux thématiques : le principe de remboursement et le retour intermédiaire. Le retour intermédiaire étant prioritaire, c'est cette thématique qui est renvoyée par les règles.

Repas-trajet : Il s'agit des conditions de remboursement des repas pris sur le trajet (dans le train ou sur une aire de repos par exemple) soit lors d'un retour intermédiaire soit lors des premier et dernier jour de stage.

Exemple : *" **QU :** L'intéressé conteste le décompte de l'ordre de mission en stage*

du 01/09 au 03/09/15 suite à 01 repas soir du 02/09/15 en secteur administratif ainsi que 02 ***repas soir trajet*** *secteur privée payant du 01/09/15 et 03/09/15 non réglés. vous joint en annexe l'attestation de stage qui stipule repas du soir onéreux par l'état.* ***RE :*** *L'administré ne peut pas obtenir l'****indemnisation du repas trajet*** *du 01/09/2015 (soir) - hors plage horaire. ",2* Ici, le moteur de règles trouve deux références à la thématique du repas trajet.

Principe de remboursement : C'est la thématique, sémantiquement et lexicalement, la plus difficile à détecter, et à définir. Cela englobe toutes les questions concernant des remboursements du stage autre que les deux premières thématiques citées ci-dessus. Par exemple, si le stagiaire doit déjeuner et diner dans le secteur privé, quel est le taux de remboursement. Pareil s'il doit se loger, se déplacer en moyens de transport entre son logement et son lieu de travail...

Exemple : *"* ***QU :*** *actuellement les personnels qui vont faire des séminaires sont codifiés en stage (stage en centre d'instruction militaire ou en stage). Pouvez-vous me dire si c'est correct car notre cellule nous a demandé de les codifier en mission.* ***RE :*** *Pour un déplacement séminaire l'ordre de mission doit être codifié en mission. En conséquence aucune attestation de stage ne sera à transmettre au CAMID -* ***indemnisation au taux mission****. ",3* Dans ce troisième exemple, le moteur de règles n'est pas assez couvrant pour trouver une thématique dans la question (le mot "mission" n'étant pas assez univoque pour déclencher à lui seul une règle). Par contre, dans la réponse, il trouve bien une représentation linguistique de la thématique du remboursement.

La disparité linguistique de cette thématique nous a posé beaucoup de problèmes à la fois lors de l'analyse et lors de l'amélioration du moteur de recherche.

Des critères exclusifs à certaines thématiques (les syntagmes *repas-trajet* ou *hébergement onéreux*, par exemple) facilitent le travail de construction de certaines règles décrivant chacune des trois thématiques.

3.2.2 le *chatbot*

Le rôle du *chatbot* est de vérifier que l'on trouve, dans la question préalablement analysée, assez d'éléments pour l'envoyer au moteur de recherche. Par contre, nous partons du principe que l'analyse s'est bien déroulée. Autrement dit, le *chatbot* n'a pas vocation à déterminer la pertinence des critères et des thématiques analysés mais seulement de les compléter, en interrogeant l'utilisateur (l'employé administratif), notamment pour lever des ambiguïtés sur certaines formes textuelles équivoques : *"déplacement régulier pendant le stage"* cela signifie-t-il *déplacement quotidien travail-domicile* ou *déplacement intermédiaire les week-ends travail-foyer familial ?* . Notre *chatbot* est un dialogue composé d'une vingtaine de branches. La sortie du *chatbot* doit être la question initiale normalisée (thématique + critères analysés seulement et opérateurs logiques (**AND ; OR ; NOT**)) corroborée des éléments indiqués par l'utilisateur au *chatbot* dans les différentes branches du dialogue. En

définitive, le *chatbot* nous a permis d'améliorer le score de l'analyse des questions notamment lorsque celle-ci génère du silence en ne détectant pas tous les critères, ou que dans la question initiale les critères ne sont pas présents alors qu'ils le devraient. Pour vérifier leur nécessité, nous avons utilisé le corpus de référence des questions-type.
Nous les avons analysées manuellement afin d'extraire des patrons sémantiques-types que le classifieur et ensuite le *chatbot* doivent retrouver pour une question inconnue.
Si la classification des questions échoue parfois à détecter la bonne thématique, le *chatbot* diminue le taux d'erreurs du moteur de recherche en complétant ou parfois en corrigeant la classification initiale. Cependant, il reste des cas où la question comporte si peu d'éléments que tout le travail incombe au robot.

3.2.3 Le moteur de recherche

Dans cette dernière brique du système, le but est de fournir un moteur de recherche performant. Pour rappel, jusqu'à maintenant nous possédons sur le papier une question analysée et normalisée en thématiques et critères suffisante pour une bonne recherche.
Ici, ce qu'il a fallu réaliser est une indexation de la documentation réglementaire déjà structurée, en la partitionnant en paragraphes, puis effectuer sur eux une analyse pour faire correspondre les différentes représentations textuelles des concepts similaires, à la fois dans ces paragraphes et dans la question de l'employé administratif, afin de posséder un moteur assez efficace pour obtenir la bonne réponse le plus souvent possible.
Enfin, nous avons réfléchi à la meilleure façon d'améliorer le moteur. Nous nous sommes rendus compte que ce qui donnait de meilleurs résultats restait la pondération des critères et des thématiques présents dans la question.
Par exemple, dans une question comme **" Dans le cadre d'un *retour intermédiaire*, le stagiaire peut-il être indemnisé de ses *repas durant le trajet*?"**, nous détectons deux thématiques. Il faut alors les hiérarchiser dans le but de fournir la réponse la plus pertinente. Dans ce cas-ci, il faut donner la priorité au **repas-trajet**, et considérer le **retour intermédiaire** comme une sous-thématique permettant d'affiner la recherche dans le moteur. Le moteur reste à ce jour la partie la plus délicate de notre système car le nombre de paragraphes est supérieur au nombre de différents types de questions que peuvent poser les militaires. De fait, plusieurs réponses sont souvent candidates pour une même question, sans que le moteur ne soit capable de trancher parmi ces candidates.

Dans la figure 3.1, nous pouvons observer la structure (simplifiée) de notre solution. Ce qu'il faut remarquer, c'est l'omniprésence du *chatbot*. En effet, il sert d'enveloppe pour l'ensemble du système afin de maintenir une interaction permanente avec l'utilisateur. Ainsi, le *chatbot* "masque" les outils et les méthodes utilisés. Nous pouvons distinguer ici les trois niveaux du système de question-réponse : l'analyse de la question, l'extraction des mots-clés et son indexation, et la recherche de la réponse.

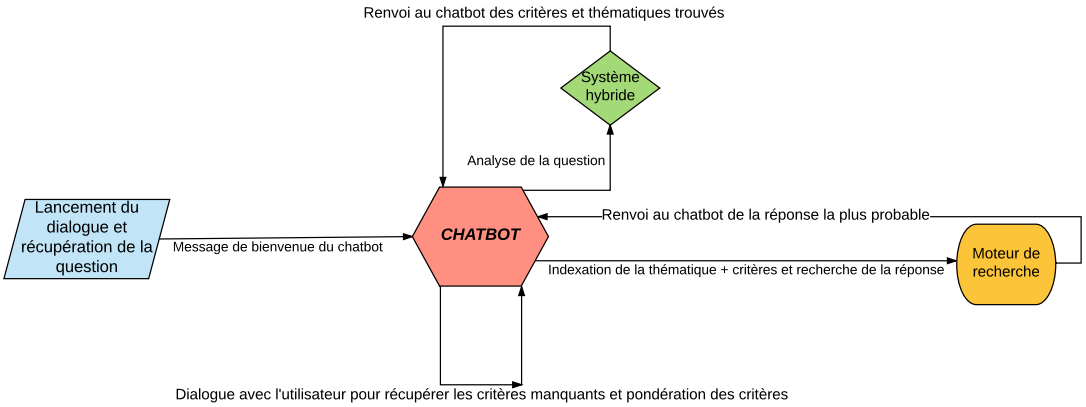


FIGURE 3.1 – Schéma simplifié de notre solution

3.3 Les moyens et méthodes utilisés

Nous avons utilisé Watson pour réaliser toutes ces tâches. Cependant, nous allons porter une attention particulière aux méthodes, en particulier en ce qui concerne l'analyse des questions et la construction du dialogue.

3.3.1 analyse de la question

Nous avons utilisé, dans un premier temps, **Watson Studio** pour créer nos lexiques et nos règles. Au bout d'un mois de travail, nous parvenions à détecter la bonne thématique dans 75% des questions. Cependant, alors que la précision augmentait, le rappel demeurait bas. De plus, notre approche par règles pouvait détecter plusieurs thématiques (voir section 3.3.3). Par contre, elle ne pouvait pas déterminer laquelle était la bonne. Notre première idée a été de créer un algorithme naïf (si repas trajet, sinon retour intermédiaire, sinon principe de remboursement, sinon rien).
Nous avons décidé d'utiliser en parallèle une approche statistique, par réseaux

de neurones (**Watson NLC**), et vérifier si elle donnait de meilleurs résultats. En moyenne, les règles étaient plus performantes.

Nous avons implémenté un algorithme semi-supervisé (approche statistique où seule une partie du corpus est annotée, et doit prédire le reste du corpus en fonction.) appelé co-apprentissage. C'est un algorithme crée par [Avrim Blum et Tom Mitchell, 1998] qui consiste à entraîner non pas un classifieur mais deux. Chaque classifieur entraîné sur une dimension différente du corpus. Le but étant que ces deux dimensions offrent de meilleurs résultats qu'une seule. Cette technique est utilisée notamment dans l'analyse de courriels avec d'excellents résultats [Svetlana Kiritchenko and Stan Matwin,2002] en divisant les courriels en deux : les objets d'un côté et le contenu de l'autre. [Wan, 2009] a utilisé le co-apprentissage pour une analyse de sentiment interlingue anglais-chinois qui a amélioré les techniques standard d'induction et de tranduction. [Nigam and Ghani, 2000a] font un comparatif avec des techniques de classification textuelles classiques et montrent comment le co-apprentissage, avec très peu de données et en divisant correctement le corpus en deux dimensions distinctes, on peut surpasser les méthodes d'apprentissage simple.
Nous avons fait de même en divisant les questions d'un côté et les réponses de l'autre. Nos résultats, pour une première approche, étaient encourageants (80% de f-mesure) malheureusement, nous n'avons tout simplement pas eu le temps de continuer les essais et les évaluations, donc ce score n'était, malheureusement, pas significatif (possibilité d'avoir eu du sur-apprentissage)). Cependant, il constitue une piste d'amélioration future pour notre système.

C'est ainsi que nous avons pensé à une combinaison des deux approches par plusieurs algorithmes notamment le vote et les moyennes. Ainsi nous possédions un système hybride, combinant la précision des règles avec la capacité de décision de l'apprentissage automatique. Nous avons également testé des solutions en libre-accès sur le Web. Nous avons utilisé la librairie python scikit learn [Pedregosa, 2011] et l'algorithme donnant les meilleurs résultats est la régression logistique.
C'est un algorithme probabiliste qui permet de prédire les rapports de dépendance entre des variables explicatives (notre texte) et des variables à expliquer (nos classes). Un modèle de régression logistique permet de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédite est supérieure à 0,5, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure à 0,5, il ne l'est pas.
L'avantage de l'algorithme est sa simplicité par rapport aux réseaux de neurones par exemple (donc beaucoup plus rapide).
Nous avons une variable à expliquer Y et un ensemble de variables explicatives (X_0,X_1,...,X_n). Or Y n'a que deux valeurs (0 ou 1) donc nous ne pouvons pas utiliser une fonction linéaire pour trouver l'espérance de Y. Nous nous servons de la fonction logistique (toujours définie dans l'intervalle]0;1[ ) afin que l'espérance de Y soit toujours soit 0 soit 1. Nous calculons la probabilité qu'un texte ait la classe Y sachant qu'il possède la variable Xi.
Pour représenter graphiquement cette fonction logistique, on peut la linéariser en utilisant le logarithme de l'évidence où p est la probabilité que Y = 1, puis la fonction $LOGIT$.

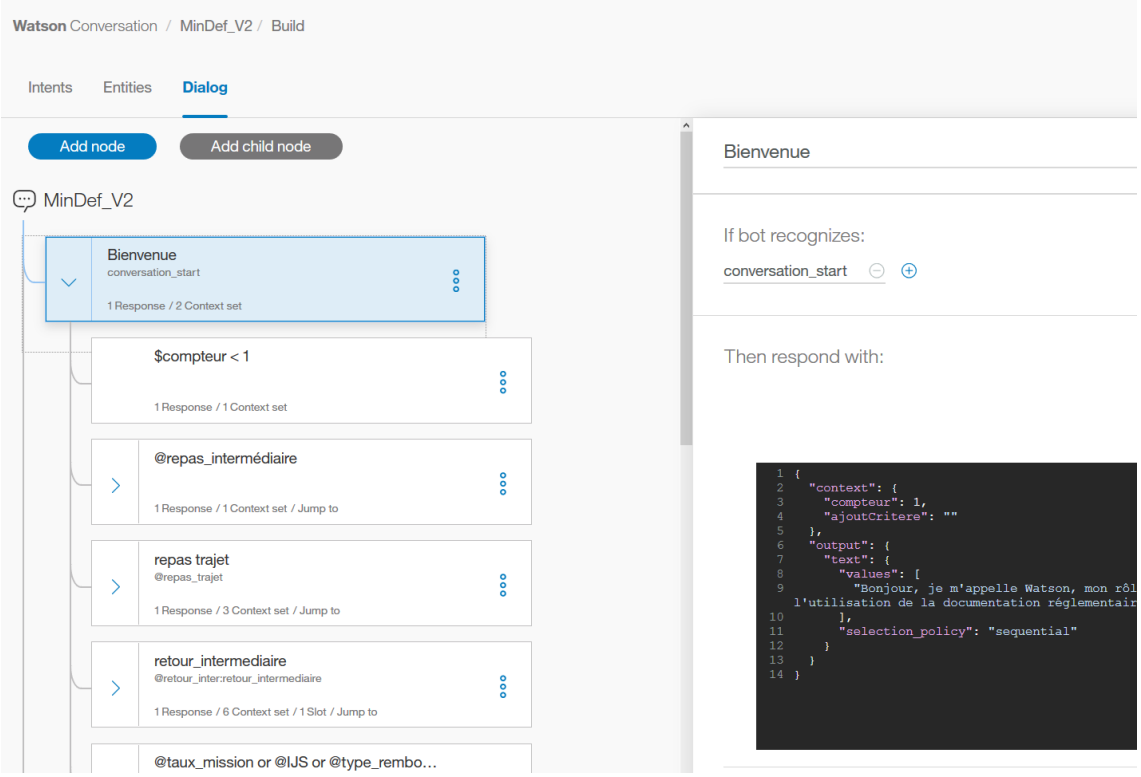


FIGURE 3.2 – première branche du dialogue

$$\text{Ev(p)} = ln(\frac{p}{1-p}) \tag{3.1}$$

Ce système améliore les règles, avec des résultats stables (qui ne changent pas significativement selon les échantillons pris pour l'évaluation). Nous en parlons dans les expérimentations.

3.3.2 le *chatbot*

Pour le *chatbot*, nous nous sommes servis de Watson Conversation.
Ce robot reçoit en entrée une question normalisée. Nous avons exporté les lexiques de notre jeu de règles et les avons intégrés à notre *chatbot* sous forme d'**entités** (section 2.2.4). Ainsi, il possède la même base que le système de règles. De plus, pour améliorer le système, nous avons créé comme **intentions** les questions-types dont nous connaissons à l'avance la thématique, et de cette façon permettre au *chatbot* d'augmenter son rappel.
Nous créons, en tout début du dialogue une variable de contexte. Cette variable de contexte est une chaîne de caractère. Sur la figure 3.1, nous pouvons voir un JSON dans lequel sont stockées en haut les variables de contexte et en bas la réponse générée par le robot lors de l'activation de la branche.

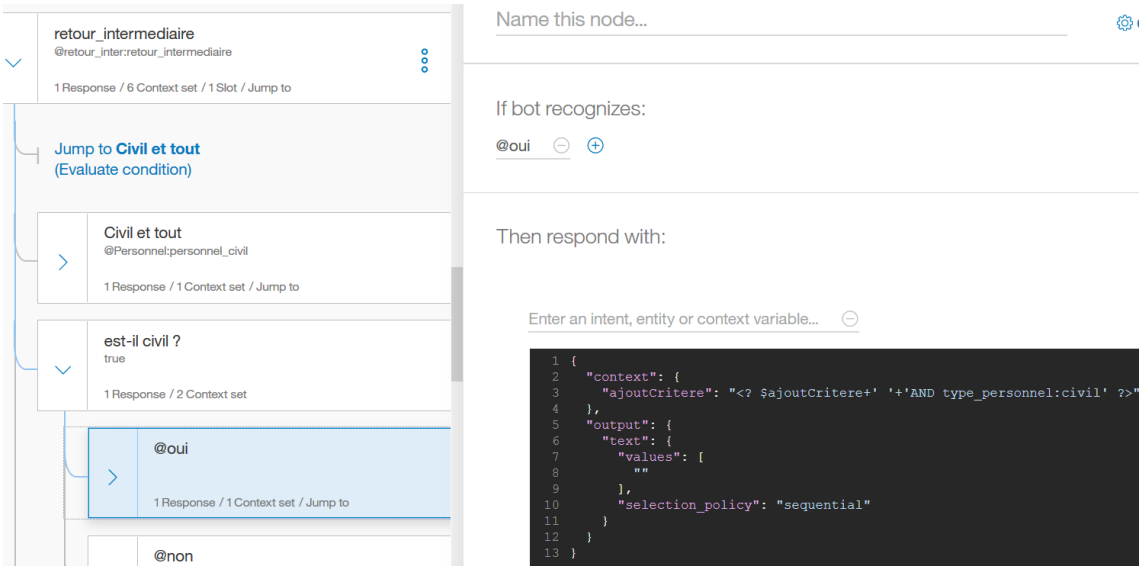


FIGURE 3.3 – une branche active du dialogue

Chaque branche du dialogue créé contient une condition correspondante à un critère indispensable au moteur de recherche, selon la thématique. Par exemple, si nous sommes dans le cadre d'un **retour intermédiaire**, il est obligatoire de savoir si le stagiaire effectue **un stage de plus ou moins de quatre semaines.**. Si dans les règles et les entités, le *chatbot* ne parvient pas à récupérer cette information ou si nous jugeons que le taux de confiance de détection ne suffit pas à valider l'information, le *chatbot* demande directement à celui qui a posé la question si son stage dure plus ou moins de quatre semaines. C'est ce que l'on appelle une branche active, car le *chatbot* dirige l'utilisateur (figure 3.2).
Après chaque branche active du dialogue, selon la réponse de l'utilisateur, la variable de contexte se remplit. Si elle n'était pas vide auparavant, on concatène ce qui existait précédemment et la nouvelle information avec un espace et le mot-outil *'AND'*. La concaténation signifie en informatique relier deux chaînes de caractères par un caractère ou une autre chaîne de caractères y compris une chaîne vide. Tout en écrivant dans la variable, il détermine grâce à un jeu de conditions, la pondération de chaque critère récupéré, avec un syntaxe compréhensible ensuite par le moteur de recherche.

Après avoir récupéré toutes les informations manquantes nécessaires, le *chatbot* concatène les thématiques et les critères détectés avant le dialogue, avec ceux de la variable de contexte. Il envoie la chaîne de caractère au moteur de recherche.

3.3.3 Le moteur de recherche

En ce qui concerne le moteur de recherche, nous avons travaillé avec Watson Explorer, et son moteur de recherche appelé **ViVisimo**. En raison de la confidentialité, nous ne pouvons pas montrer d'images du moteur de recherche. Le moteur analyse la thématique et les critères de la question préalablement normalisée par le *chatbot*. Il calcule la pondération de chaque composante afin d'extraire la meilleure réponse. Nous avons établi quatre niveaux de pondération, allant de un à dix, par ordre

TABLE 3.1 – récapitulatif des pondération

1	critères peu ou prou discriminants
3	critères discriminants
5	critères majeurs ou sous-thématiques
10	critères essentiels ou thématiques

d'importance (tableau 3.1). Ce sont ces pondérations qui affinent les paramètres de recherche. Ils sont définis au niveau du *chatbot* et calculés dans le moteur de recherche.

3.4 Les imprévus et les limites

Ce travail a toutefois été marqué par plusieurs situations imprévues. Au départ, l'objectif était de réaliser un *chatbot* capable à la fois d'analyser la question mais aussi de fournir la bonne réponse. Il nous fallait donc également intégrer la documentation réglementaire, d'une façon ou d'une autre dans le robot. Pourtant, cet objectif a dévié et c'est finalement un moteur de recherche qui était attendu. Ce qui nous a contraint à repenser notre méthodologie : nous pensions au départ faire du robot la brique finale de notre système.

Ensuite, nous y revenons en détails dans le chapitre 4, nous disposions d'un corpus de couples (questions ;réponses). Une de nos idées de départ était d'intégrer ces réponses, écrites par des employés administratifs, à la documentation réglementaire et ainsi faciliter le lien entre question et documentation, car les réponses étaient sémantiquement et linguistiquement proches des questions, contrairement à la documentation. Malheureusement, le temps nous a manqué pour mettre en place cette application, qui à notre avis, pourrait améliorer, à l'avenir, les résultats.

Enfin, pour parler des contraintes matérielles, notre corpus de départ se constituait de quatre cents question-réponse. C'est très peu pour mettre en place un système d'apprentissage statistique. Nous pensons qu'avec plus de matière nous pouvons obtenir de bien meilleurs résultats. Cependant, nous avons trouvé dix-sept thématiques différentes. La quantité de données nécessaire pour créer un système à dix-sept variables à expliquer est énorme. En effet, la quantité de données nécessaire n'est pas proportionnelle au nombre de classes : si nous avons vingt textes et deux catégories (classification binaire), nous aurons de bien meilleurs résultats que si nous avons deux cent textes et vingt catégories.
Par ailleurs, une autre contrainte était l'utilisation de Watson. Ce qui fait que notre capacité d'analyse et de recherche d'information ne pouvait pas s'étendre au-delà de celle de Watson. Or une analyse syntaxique aurait aussi pu nous permettre d'obtenir de meilleurs résultats. De même, dans **NLC**, l'outil de classification automatique, d'une part nous ne pouvions paramétrer l'algorithme (*tuning* d'algorithme), d'autre part, la limite de taille de chaque texte étant de 1024 caractères, très souvent nous devions opérer une troncature arbitraire, ou manuelle, qui provoquait dans un cas un manque d'information parfois pénalisant, et dans l'autre un biais car les données étaient choisies par l'humain. Pour y remédier nous avons créé quelques règles

simplifiant automatiquement certains textes trop longs.

3.5 Conclusion

En définitive, ce qu'il faut retenir de ce chapitre c'est l'utilisation des trois briques essentielles de notre système : l'analyse des questions par règles et l'apprentissage automatique, l'agent conversationnel et sa façon d'interagir avec l'humain, et le moteur de recherche permettant d'indexer la question et de fournir la réponse.
Nous avons ici résolu partiellement notre problématique de départ : Comment savoir si une question est bien posée ou incomplète, et dans quelle mesure pouvons-nous la rectifier ?

Deuxième partie

Expérimentations

CHAPITRE 4

CORPUS

Sommaire

4.1 Introduction

Dans ce chapitre, nous décrivons nos ressources, les traitements que nous avons appliqués au corpus ainsi que nos premières expériences.
Nous nous attardons ici sur le premier socle de notre système, l'analyse de la question car elle reste la partie essentielle du système et qui démontre l'aspect cognitif de notre solution.
Enfin nous parlons des premiers résultats, des expériences consécutives à ces résultats et de la façon dont cela a modifié notre manière de travailler afin d'obtenir des scores encourageants.

4.2 Le corpus

Dans cette section, nous expliquons comment était le corpus au départ, et comment nous l'avons manipulé afin de le rendre exploitable, notamment pour l'apprentissage automatique.

4.2.1 Acquisition et appropriation du corpus

Nous possédions un corpus inital composé de dix mille couples (questions ;réponses). Il s'agissait de questions posées par un employé administratif à son supérieur, et la réponse de ce dernier.

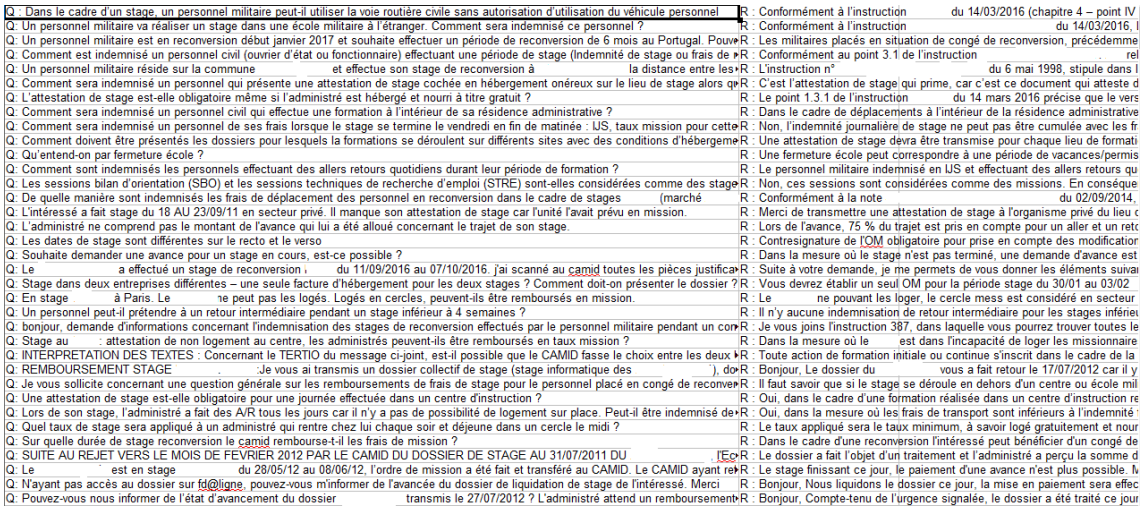
Notre rôle est non pas de répondre au militaire à la place de l'employé administratif, mais d'assister l'employé administratif dans la compréhension de la question du militaire. Autrement dit, c'est l'employé administratif qui pose la question au système. L'employé interprète la question du militaire, et s'il ne sait pas y répondre, il interroge notre moteur de recherche en simplifiant la question du militaire (par exemple, *stage du 5 mai au 8 août* devient *stage supérieur à quatre semaines*). Nous remplaçons de fait l'expert humain, supérieur hiérarchique de l'employé administratif, qui était chargé de l'aider à répondre à la question initiale du militaire.

Cela nous a posé deux problèmes : d'abord, notre corpus se compose de questions des employés à d'autres employés. De fait, le niveau de langage peut varier beaucoup d'un employé à l'autre, pour la question comme pour la réponse. Nous devons donc adapter nos règles non seulement en fonction du corpus de questions, mais aussi du vocabulaire du personnel administratif.
De plus, comme il s'agit d'un service administratif s'occupant de tous les problèmes d'un militaire, les verbatims peuvent contenir plusieurs questions à la fois, dont certaines ne montrant aucun rapport entre elles : *"' Je voulais savoir si un stagiaire est remboursé au taux de mission s'il effectué son stage de 2 mois dans un secteur privé. Est-ce que lors de ses retours intermédiaires il peut se faire indemniser ses repas-trajet ? "'*

Après ces premières observations, nous avons décidé de créer des thématiques en fonction des questions que nous avions regroupées. Ce travail de regroupement par thématique a été réalisé manuellement. Au total, nous comptions dix-sept thématiques. Nous avons considéré comme une thématique, un groupe de plus de quinze questions sémantiquement proches. Il nous semble important ici de préciser que ce groupement s'est effectué en fonction, et uniquement en fonction, de la question. La réponse a été ignorée.
Dans un premier temps, nous avons tenté de réaliser un partitionnement automatique des données (ou *clustering* en anglais). C'est un procédé d'apprentissage automatique non-supervisé (sans classification humaine préalable des données) [Jain et al., 1999], [EstivillCastro, 2002] dans lequel le système regroupe les données par grappes selon leur similarité, en sachant ou pas par avance le nombre de grappes attendues. Les résultats se sont révélés décevants.

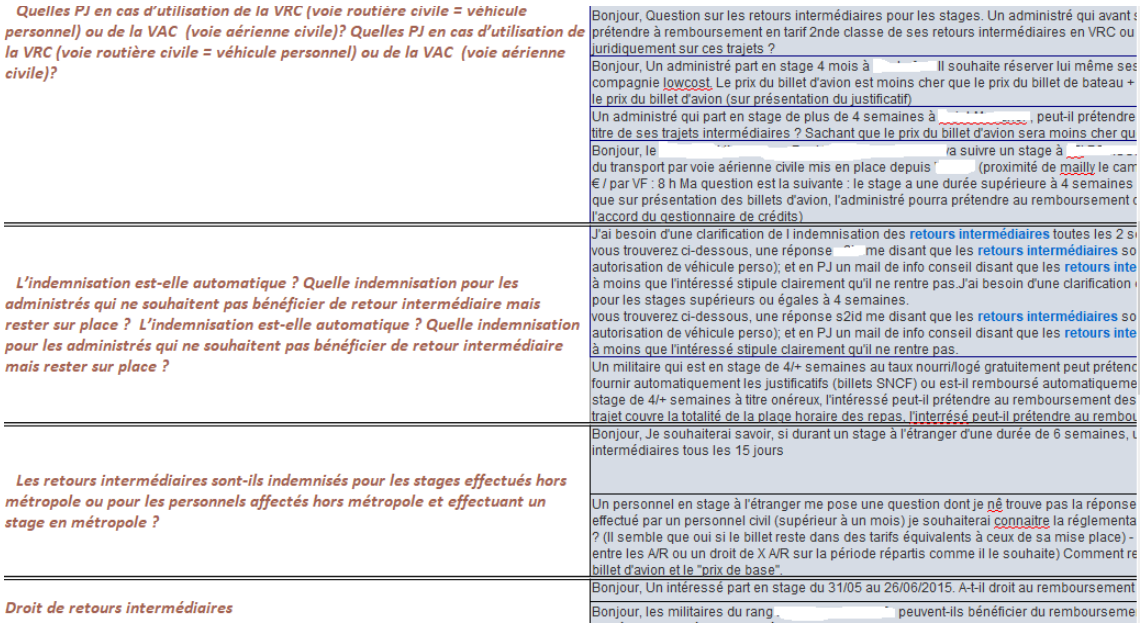
En définitive, comme évoqué dans le chapitre précédent, nous en avons retenu trois. Ces trois thématiques possédaient un peu moins de quatre cents exemples de questions-réponses (370) Cela représente un peu moins de 50 000 mots(figure 4.1).

Nous possédions des questions-types artificielles correspondant à ces thématiques. Ces questions-types se présentent de la façon suivante : le thème, pour chaque thème des sous-thèmes, pour chaque sous-thème des exemples textuels réels. Il va de soi que les questions-types ne font pas partie du corpus des 370 questions, cela biaiserait notre étude à l'heure d'évaluer le système d'analyse des questions (la figure 4.2 montre un extrait des questions-types pour le **retour intermédiaire**).



Q :	R :
Dans le cadre d'un stage, un personnel militaire peut-il utiliser la voie routière civile sans autorisation d'utilisation du véhicule personnel	Conformément à l'instruction du 14/03/2016 (chapitre 4 – point IV
Un personnel militaire va réaliser un stage dans une école militaire à l'étranger. Comment sera indemnisé ce personnel ?	Conformément à l'instruction du 14/03/2016, l
Un personnel militaire est en reconversion début janvier 2017 et souhaite effectuer un période de reconversion de 6 mois au Portugal. Pouv	Les militaires placés en situation de congé de reconversion, précédemme
Comment est indemnisé un personnel civil (ouvrier d'état ou fonctionnaire) effectuant une période de stage (Indemnité de stage ou frais de	Conformément au point 3.1 de l'instruction rel
Un personnel militaire réside sur la commune et effectue son stage de reconversion à la distance entre les	L'instruction n° du 6 mai 1998, stipule dans l
Comment sera indemnisé un personnel qui présente une attestation de stage cochée en hébergement onéreux sur le lieu de stage alors qu	C'est l'attestation de stage qui prime, car c'est ce document qui atteste d
L'attestation de stage est-elle obligatoire même si l'administré est hébergé et nourri à titre gratuit ?	Le point 1.3.1 de l'instruction du 14 mars 2016 précise que le vers
Comment sera indemnisé un personnel civil qui effectue une formation à l'intérieur de sa résidence administrative ?	Dans le cadre de déplacements à l'intérieur de la résidence administrative
Comment sera indemnisé un personnel de ses frais lorsque le stage se termine le vendredi en fin de matinée : IJS, taux mission pour cette	Non, l'indemnité journalière de stage ne peut pas être cumulée avec les fr
Comment doivent être présentés les dossiers pour lesquels la formations se déroulent sur différents sites avec des conditions d'hébergeme	Une attestation de stage devra être transmise pour chaque lieu de formati
Qu'entend-on par fermeture école ?	Une fermeture école peut correspondre à une période de vacances/permis
Comment sont indemnisés les personnels effectuant des allers retours quotidiens durant leur période de formation ?	Le personnel militaire indemnisé en IJS et effectuant des allers retours qu
Les sessions bilan d'orientation (SBO) et les sessions techniques de recherche d'emploi (STRE) sont-elles considérées comme des stage	Non, ces sessions sont considérées comme des missions. En conséque
De quelle manière sont indemnisés les frais de déplacement des personnel en reconversion dans le cadre de stages (marché	Conformément à la note du 02/09/2014,
L'intéressé a fait stage du 18 AU 23/09/11 en secteur privé. Il manque son attestation de stage car l'unité l'avait prévu en mission.	Merci de transmettre une attestation de stage à l'organisme privé du lieu d
L'administré ne comprend pas le montant de l'avance qui lui a été alloué concernant le trajet de son stage.	Lors de l'avance, 75 % du trajet est pris en compte pour un aller et un ret
Les dates de stage sont différentes sur le recto et le verso	Contresignature de l'OM obligatoire pour prise en compte des modification
Souhaite demander une avance pour un stage en cours, est-ce possible ?	Dans la mesure où le stage n'est pas terminé, une demande d'avance est
Le a effectué un stage de reconversion du 11/09/2016 au 07/10/2016. j'ai scanné au camid toutes les pièces justifica	Suite à votre demande, je me permets de vous donner les éléments suivar
Stage dans deux entreprises différentes – une seule facture d'hébergement pour les deux stages ? Comment doit-on présenter le dossier ?	Vous devrez établir un seul OM pour la période stage du 30/01 au 03/02
En stage à Paris. Le ne peut pas les logés. Logés en cercles, peuvent-ils être remboursés en mission.	Le ne pouvant les loger, le cercle mess est considéré en secteur
Un personnel peut-il prétendre à un retour intermédiaire pendant un stage inférieur à 4 semaines ?	Il n'y aucune indemnisation de retour intermédiaire pour les stages inférieu
bonjour, demande d'informations concernant l'indemnisation des stages de reconversion effectués par le personnel militaire pendant un con	Je vous joins l'instruction 387, dans laquelle vous pourrez trouver toutes le
Stage au : attestation de non logement au centre, les administrés peuvent-ils être remboursés en taux mission ?	Dans la mesure où le est dans l'incapacité de loger les missionnaire
INTERPRETATION DES TEXTES : Concernant le TERTIO du message ci-joint, est-il possible que le CAMID fasse le choix entre les deux	Toute action de formation initiale ou continue s'inscrit dans le cadre de la
REMBOURSEMENT STAGE . Je vous ai transmis un dossier collectif de stage (stage informatique des), do	Bonjour, Le dossier du vous a fait retour le 17/07/2012 car il y
Je vous sollicite concernant une question générale sur les remboursements de frais de stage pour le personnel placé en congé de reconve	Il faut savoir que si le stage se déroule en dehors d'un centre ou école mil
Une attestation de stage est-elle obligatoire pour une journée effectuée dans un centre d'instruction ?	Oui, dans le cadre d'une formation réalisée dans un centre d'instruction re
Lors de son stage, l'administré a fait des A/R tous les jours car il n'y a pas de possibilité de logement sur place. Peut-il être indemnisé de	Oui, dans la mesure où les frais de transport sont inférieurs à l'indemnité
Quel taux de stage sera appliqué à un administré qui rentre chez lui chaque soir et déjeune dans un cercle le midi ?	Le taux appliqué sera le taux minimum, à savoir logé gratuitement et nour
Sur quelle durée de stage reconversion le camid rembourse-t-il les frais de mission ?	Dans le cadre d'une reconversion l'intéressé peut bénéficier d'un congé de
SUITE AU REJET VERS LE MOIS DE FEVRIER 2012 PAR LE CAMID DU DOSSIER DE STAGE AU 31/07/2011 DU (Ec	Le dossier a fait l'objet d'un traitement et l'administré a perçu la somme d
Le est en stage du 28/05/12 au 08/06/12, l'ordre de mission a été fait et transféré au CAMID. Le CAMID ayant re	Le stage finissant ce jour, le paiement d'une avance n'est plus possible. M
N'ayant pas accès au dossier sur fd@ligne, pouvez-vous m'informer de l'avancée du dossier de liquidation de stage de l'intéressé. Merci	Bonjour, Nous liquidons le dossier ce jour, la mise en paiement sera effec
Pouvez-vous nous informer de l'état d'avancement du dossier transmis le 27/07/2012 ? L'administré attend un remboursement	Bonjour, Compte-tenu de l'urgence signalée, le dossier a été traité ce jour

FIGURE 4.1 – extrait du corpus initial de 370 questions-réponses (les passages confidentiels sont effacés)



Quelles PJ en cas d'utilisation de la VRC (voie routière civile = véhicule personnel) ou de la VAC (voie aérienne civile)? Quelles PJ en cas d'utilisation de la VRC (voie routière civile = véhicule personnel) ou de la VAC (voie aérienne civile)?	Bonjour, Question sur les retours intermédiaires pour les stages. Un administré qui avant s prétendre à remboursement en tarif 2nde classe de ses retours intermédiaires en VRC ou juridiquement sur ces trajets ?
	Bonjour, Un administré part en stage 4 mois à Il souhaite réserver lui même ses compagnie lowcost. Le prix du billet d'avion est moins cher que le prix du billet de bateau + le prix du billet d'avion (sur présentation du justificatif)
	Un administré qui part en stage de plus de 4 semaines à , peut-il prétendre titre de ses trajets intermédiaires ? Sachant que le prix du billet d'avion sera moins cher qu
	Bonjour, le /a suivre un stage à du transport par voie aérienne civile mis en place depuis (proximité de mailly le cam € / par VF : 8 h Ma question est la suivante : le stage a une durée supérieure à 4 semaines que sur présentation des billets d'avion, l'administré pourra prétendre au remboursement d l'accord du gestionnaire de crédits)
L'indemnisation est-elle automatique ? Quelle indemnisation pour les administrés qui ne souhaitent pas bénéficier de retour intermédiaire mais rester sur place ? L'indemnisation est-elle automatique ? Quelle indemnisation pour les administrés qui ne souhaitent pas bénéficier de retour intermédiaire mais rester sur place ?	J'ai besoin d'une clarification de l indemnisation des **retours intermédiaires** toutes les 2 s vous trouverez ci-dessous, une réponse me disant que les **retours intermédiaires** so autorisation de véhicule perso); et en PJ un mail de info conseil disant que les **retours inte** à moins que l'intéressé stipule clairement qu'il ne rentre pas.J'ai besoin d'une clarification pour les stages supérieurs ou égales à 4 semaines. vous trouverez ci-dessous, une réponse s2id me disant que les **retours intermédiaires** so autorisation de véhicule perso); et en PJ un mail de info conseil disant que les **retours inte** à moins que l'intéressé stipule clairement qu'il ne rentre pas.
	Un militaire qui est en stage de 4/+ semaines au taux nourri/logé gratuitement peut prétend fournir automatiquement les justificatifs (billets SNCF) ou est-il remboursé automatiqueme stage de 4/+ semaines à titre onéreux, l'intéressé peut-il prétendre au remboursement des trajet couvre la totalité de la plage horaire des repas, l'interrésé peut-il prétendre au rembou
Les retours intermédiaires sont-ils indemnisés pour les stages effectués hors métropole ou pour les personnels affectés hors métropole et effectuant un stage en métropole ?	Bonjour, Je souhaiterai savoir, si durant un stage à l'étranger d'une durée de 6 semaines, intermédiaires tous les 15 jours
	Un personnel en stage à l'étranger me pose une question dont je ne trouve pas la réponse effectué par un personnel civil (supérieur à un mois) je souhaiterai connaitre la réglementa ? (Il semble que oui si le billet reste dans des tarifs équivalents à ceux de sa mise place) - entre les A/R ou un droit de X A/R sur la période répartis comme il le souhaite) Comment re billet d'avion et le "prix de base".
Droit de retours intermédiaires	Bonjour, Un intéressé part en stage du 31/05 au 26/06/2015. A-t-il droit au remboursement
	Bonjour, les militaires du rang peuvent-ils bénéficier du rembourseme

FIGURE 4.2 – extrait des questions-types dans la thématique du retour intermédiaire

4.2.2 Préparation et classification des données

Dans une première partie de ce travail, nous n'avons pas considéré opportun d'exploiter les réponses dans le corpus. Nous avons donc créé un premier corpus avec les 370 questions.

Nous en avons sélectionné aléatoirement 300 qui nous ont servi à créer les lexiques et ainsi les règles. Les 70 restantes étant dédiées à la future évaluation de l'approche symbolique dans Watson Studio.
Toutefois, si nous voulions pouvoir évaluer les règles, nous avions quand même besoin d'une classification manuelle 100% humaine afin de pouvoir tester Watson.
Nous avons alors décidé de la nomenclature suivante : pour chacune des 370 questions nous attribuions une et une seule classe. En sachant que le **repas trajet** était prioritaire sur le **retour intermédiaire**, lui-même prioritaire sur **le principe de remboursement**(figure 4.3).

Quatre annotateurs ont classifié le corpus de 370 questions. Nous obtenons un score inter-annotateurs de 0.5 (coefficient de Kappa) en moyenne. C'est un score assez médiocre. Cependant, si l'on tient compte du fait que nous, les annotateurs, ne connaissons ni le corpus, ni le métier, ce score faible est à nuancer. Par classe, la thématique **principe d'indemnisation** reste la plus problématique car l'accord de Kappa tombe à 0.2 pour cette seule catégorie.

Nous avons semi-aléatoirement séparé le corpus en deux sous-corpus, pour l'apprentissage et pour l'évaluation. Nous gardons 25% du corpus pour l'évaluation (environ 90 questions) et le reste pour l'apprentissage. Semi-aléatoirement signifie que nous avons sélectionné à l'aveugle 90 questions tout en respectant les proportions de chaque classe dans le corpus entier. Cela s'explique par le fait que les **repas-trajet** sont la catégorie la moins bien représentée, nous devions être certains que le corpus d'apprentissage et celui d'évaluations contenaient des exemples de cette thématique.
Ensuite, nous avons créé avec ce corpus de test une *collection* sur **Watson Content Analytics** pour tester nos lexiques et nos règles. Pour ce faire, nous avons écrit un script python en utilisant un appel API (*Application Programming Interface*) vers Watson Content. Nous avons ensuite utilisé scikit learn pour évaluer nos résultats. Avec cette librairie, nous avons effectué le calcul de la précision du rappel et de la f-mesure (voir annexe). La f-mesure globale est de 0.77.

Le rôle de ce nettoyage était double. D'une part, afin de faciliter le travail de Watson, des phrases plus courtes et avec moins de bruit s'imposaient. D'autre part, nous le verrons dans la section 4.2.3, nous l'avions évoqué, pour l'apprentissage automatique, **Watson NLC** nous contraint à n'utiliser des textes dont la longueur ne dépasse pas les 1024 caractères. Ce nettoyage a permis de régler ce problème dans la plupart des questions. Pour les quelques très longues questions restantes, nous avons décidé de pratiquer un découpage arbitraire en ne conservant que les premiers 1024 caractères, quels que soient les suivants.

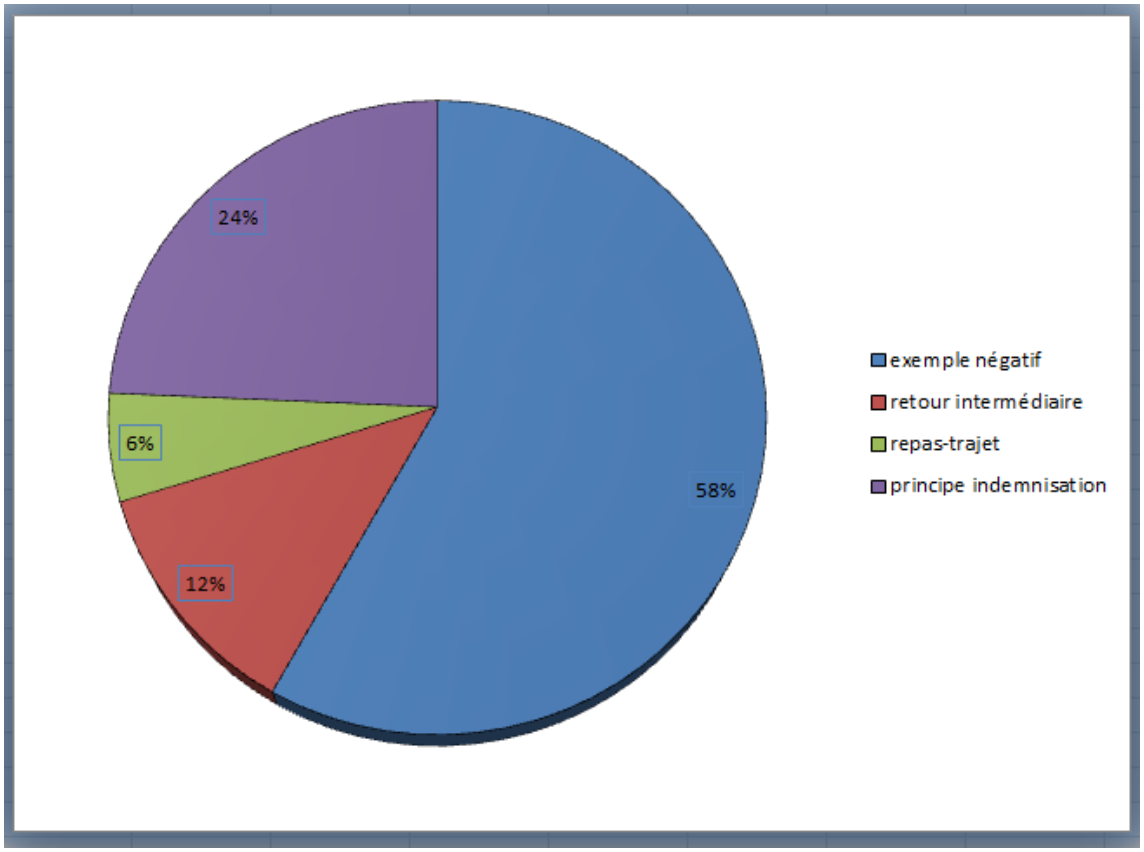


FIGURE 4.3 – répartition des occurrences de chaque thématique

4.2.3 Approche par règle

Cette étape consiste en l'appropriation à la fois du corpus et du métier. Ces règles sont la formalisation par des automates de notre classification manuelle.
Dans un premier temps, il a été essentiel de créer un lexique du métier. Nous avons utilisé l'outil Watson Studio pour les bâtir.
Nous avons décidé de construire nos dictionnaires au niveau le plus unitaire possible. Par exemple, au lieu de regrouper dans un seul et même dictionnaire tous les moyens de transport, nous avons créé un dictionnaire pour chaque type de transport, et les avons ensuite regroupés sous une règle plus générale appelée **"Transports"**. Cela nous a permis de pouvoir manipuler un transport spécifique lorsque nous en avions le besoin (par exemple **"Train"**, et les transports de façon globale. De cette façon, nous avons évité de possibles effets de bord avec la règle générale. Un autre avantage de cette granularité des dictionnaires est qu'elle permet d'y rajouter des synonymes n'importe quand, en particulier lorsque nous rencontrons une variante ou une déviance orthographique d'un mot.
Dans un second temps, lorsque les dictionnaires ont été tous créés et que le corpus a été quadrillé, nous avons construit les règles. Nous avons utilisé deux approches : soit les outils présents dans Watson, soit les expressions régulières.

Les fonctions dans Watson Studio

Watson Studio annote automatiquement le document que nous lui donnons. Plus précisément, il le segmente et il effectue une analyse morphologique et une lemmatisation. Ainsi, nous pouvons nous servir du lemme de chaque mot ou de sa catégorie grammaticale.
Par exemple, dans une phrase comme Le personnel militaire en stage du 15/02 au 16/04 a-t-il le droit de rentrer chez lui tous les 15 jours? ", nous trouvons plusieurs déclenchements de dictionnaires : **personnel militaire**, lui-même composé de **"personnel"** et **"militaire"** (car nous pouvons trouver **"personnel civil"**). Nous trouvons **"stage"**, **"droit"** et
textcolororangetextbf"rentrer chez lui", une règle, elle-même composée du dictionnaire **"rentrer"** et d'une règle *"chez + pronom personnel"*.
Nous faisons remarquer ici que selon les cas de figure, certains dictionnaires sont définis par le paradigme entier d'un mot (en général les noms communs, et certains verbes comme **"voyager"**) par contre il nous faut faire attention à ne pas utiliser le lemme de certains mots trop équivoques comme **"rentrer"**. Cela produit un bruit imprévu **"j'ai voulu rentrer le tout ce jour mais cela m'est impossible car OM inexistant."**
A l'inverse, nous pouvons créer un dictionnaire contenant des mots différents synonymes et utilisés exactement dans le même contexte : "**remboursement"** et **"indemnisation"** sont inclus dans le dictionnaire **"remboursement"**.
"Tous les 15 jours" déclenche une sous-règle appelée *retour*. C'est une sous-règle car elle n'est jamais implémentée par notre analyseur, elle sert à déclencher une autre règle appelée*retour intermédiaire.* Celle-ci s'active lorsque l'analyseur trouve une instance du dictionnaire **"rentrer"** et la règle *"retour"*. Cette méthode provoque un peu de silence car il existe d'autres formulations pour exprimer le retour régulier que nous n'avons pas trouvées ou auxquelles nous n'avons pas songé. Elle permet de limiter le bruit car au départ elle était trop permissive. Dans *retour*, nous trouvons aussi d'autres représentations linguistiques proches :
- "toutes les deux semaines"
- "Deux fois par mois"
- "un week-end sur deux"...

Après avoir créé ces règles de surface, il nous faut les tester. Nous lançons donc une nouvelle annotation du corpus mais cette fois avec les nouvelles règles sémantiques. Si nous trouvons qu'elles ne sont pas assez couvrantes, nous les améliorons en formalisant de nouveaux exemples textuels que nous auraient échappé.
Cela devient plus compliqué lorsque nous trouvons du bruit. Dans ce cas-là, nous avons deux solutions.
La première consiste à *annuler* une règle dans le cas précis du faux positif trouvé. Par exemple :**"Le personnel militaire rentre au mess le soir depuis deux semaines, il ne reste plus sur place"**. Il nous faut alors formaliser une *anti-règle* pour empêcher que notre analyseur n'annote cet exemple.
Si cela ne suffit pas, c'est que nous avons fait une erreur de cohérence au niveau des dictionnaires et cela nous contraint à retravailler en profondeur la structure de nos ressources. Cela arrive lorsque l'on crée un dictionnaire trop permissif ou que deux dictionnaires entrent en conflit car il sont utilisés dans deux règles non compatibles.
Nous avons utilisé également les agrégats. Ce sont des règles efficaces quand nous travaillons au niveau de la phrase. Les agrégats possèdent la caractéristique d'igno-

rer tous les *tokens* (mots ou ponctuation) qui ne sont pas annotés sémantiquement (non détectée en tant qu'instance d'un dictionnaire ou d'une règle). Ainsi, si deux règles ou dictionnaires sont présents dans une même phrase, dans un certain ordre, mais que l'on ne tient pas compte de ce qu'il peut y avoir entre les deux, la fonction agrégat se révèle puissante. Cependant, nous l'avons très peu utilisée car elle reste très dangereuse, dans la mesure où elle peut parfois générer un agrégat entre deux règles qui ne sont pas du tout corrélées.
Enfin, nous pouvons rendre certains *tokens* facultatifs, ou répétables plusieurs fois. Par exemple : textbf'Lors de son trajet de dernier jour, l'administré peut-il être remboursé de son repas pris durant ce trajet ?" Ici, nous n'avons pas besoin de l'information **"pris durant ce.."**. De même, si la proposition relative n'avait pas été introduite par un participe mais par un pronom, celui-ci n'aurait pas été nécessaire, comme tout *token* incrusté entre "**repas"** et **"trajet"**. Nous ajoutons comme complément à la règle du **"repas-trajet"** le fait qu'un certain nombre de *tokens* intermédiaires peuvent, ou pas, se trouver entre les instances du dictionnaire **"repas"** et **"trajet"**. Dans la mesure où nous ne pouvons pas établir une liste exhaustive de tous les mots intercalés entre les deux (participiale, relative, prépositionnelle...) nous décidons dans cette situation d'établir un nombre arbitraire de *tokens* pouvant se trouver entre les deux dictionnaires. De façon générale, il est rare de trouver plus de six mots entre les deux instances.

Les expressions régulières

Les fonctions de Watson montrent quand même leurs limites lorsque nous devons affiner certaines règles. Par exemple, l'analyse morphologique reste superficielle. En effet, "il" et "nous" seront considérés tous deux comme des pronoms personnels mais il n'existe aucune distinction de genre, de nombre ni de personne entre eux. Lorsque nous voulons créer une règle dans laquelle il existe un pronom personnel de première personne, nous sommes forcés d'écrire une expression régulière grossière de la forme "je OU nous".
Ensuite, pour repérer les dates, comme dans l'exemple fourni dans la section précédente, il est bien plus pratique d'écrire une expression régulière pour capter toutes les variations (formats de date, présence ou absence de l'année, année écrite avec deux ou quatre chiffres, jours et mois séparés par des '/', des '-'...). Nous ne pouvons pas créer une expression universelle qui fonctionne dans toutes les dates. Par contre, un des chantiers à l'avenir est de récupérer suffisamment de dates et d'entraîner l'outil d'apprentissage automatique Watson Knowledge Studio sur celles-ci. Il est fort possible qu'ainsi nous puissions réduire le silence. Les codes (numéros de dossier..) sont un autre exemple où WKS pourrait être efficace.

Résultats des règles

Nous avons évalué ces règles. Les résultats sont retranscrits dans le tableau (4.1).

Nous devions améliorer ces résultats sinon le moteur de recherche risquait de se montrer peu performant. Nous avons dans un premier temps effectué des essais en

TABLE 4.1 – Evaluation des trois sous corpus par les règles (f-mesure)

question	0.77
réponse	0.73
question+réponse	0.72

regardant si les réponses, elles, amélioraient les résultats, et si le couple (question ;réponse) aussi. Ils obtiennent respectivement 0.73 et 0.72 de f-mesure.
Nos règles ne suffisaient plus, il nous fallait obtenir par un autre moyen l'amélioration des résultats. Nous avons alors créé un autre jeu de règles dans **Watson Studio** destiné non pas à classifier mais à nettoyer les questions. Le reste du nettoyage nous l'avons effectué à la main.
Par nettoyage, nous entendons l'élimination des mots ou des phrases inutiles et/ou susceptibles de tromper Watson, comme les salutations et les signatures, ou les multi-questions en ne gardant que celle qui nous intéresse.

4.2.4 Apprentissage automatique NLC et libre accès

L'apprentissage automatique est une méthode statistique permettant à un système de devenir "intelligent". Par intelligence, nous comprenons un système qui, pour une tâche *a priori* impossible donnée à un instant A, devient faisable à un instant B car entretemps le système a appris à la réaliser.
Dans notre travail, un système d'apprentissage automatique est un système capable de classifier correctement une question qu'on lui pose, et à s'améliorer au fil du temps, selon les trois thématiques sélectionnées au début de notre étude (et le complémentaire de la somme de ces trois-là : l'exemple négatif).

Apprentissage et l'évaluation

L'apprentissage d'un système se fait de la façon suivante : nous lui fournissons notre nomenclature et plusieurs exemples de chaque thématique pour qu'il s'entraîne à les reconnaître. Il existe plusieurs méthodes d'apprentissage automatique plus ou moins efficaces selon la nature des données et de la tâche demandée. Par exemple, certains algorithmes fonctionnent mieux selon la quantité de données, s'il s'agit de chiffres ou de textes, d'images, si l'on veut faire de la classification, ou de la reconnaissance... De plus, l'avantage de l'apprentissage automatique est son affranchissement des règles. En effet, il établit des corrélations et tire des conclusions par calculs de similarité ou de régression mais ne se cantonne pas à des formes textuelles limitées pour s'entraîner, à l'inverse des règles. Cependant, il n'existe pas de règle universelle permettant d'affirmer que tel algorithme fonctionne toujours bien dans telle situation. De plus, ils sont paramétrables : l'analyse peut posséder une certaine profondeur (nombre de fois que l'algorithme "lit" les données pour être sûr de bien les apprendre), un certain calibre (nombre de mots ignorés ou pondérés selon leur importance)... Pour faire une comparaison, ce serait comme chercher la combinaison d'un coffre-fort en tâtonnant avec un stéthoscope. C'est pourquoi nous en avons testé plusieurs que nous citons à la fin de cette section.

L'évaluation, elle, s'effectue après que le système s'est suffisamment entraîné. Cela peut être décidé par celui qui l'entraîne, en ordonnant qu'après un certain temps ou un certain nombre de lectures l'apprentissage soit terminé, ou par le

système lui-même.

NLC

Watson NLC utilise les réseaux de neurones artificiels pour effectuer son apprentissage automatique. C'est un ensemble d'algorithmes dont la conception est à l'origine très schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques. Ils sont utilisés en intelligence artificielle car ils permettent de s'entraîner et de créer des mécanismes d'apprentissage indépendants de ceux imaginés par les êtres humains.
De façon plus formelle, un réseau de neurones à une ou plusieurs couches prend en entrée une liste de couples question-classe (q ;c) où $\vec{q}$ est un vecteur de variables aléatoires (dans notre cas les mots d'une question : q_0,q_1,...,q_n).
A chaque couche, le réseau connecte les neurones avec des poids (aussi appelés coefficients synaptiques) selon la forte connexion entre deux variables (w_0,w_1,...,w_n), et y introduit un biais b. Il calcule la sortie et compare avec celle attendue (c).Si ce n'est pas le cas, il met à jour les poids. Pour évaluer la sortie, le réseau fait la somme de ces variables et de leur poids et compare le résultat par rapport au seuil θ que l'on a fixé.

$$\text{Résultat} = \sum_{i=0}^{n} q_i w_i \tag{4.1}$$

Le résultat r, est appelé un potentiel d'activation. Selon la fonction de transfert de Heavyside (mais il en existe d'autres comme la fonction Sigmoïde) (si r > 0 sortie = 1 ; si $r \leq 0$ sortie = 0), selon le signe du résultat, le réseau a correctement appris, ou pas. Le réseau de neurones s'arrête dès lors que la sortie est à 1 ou qu'il a effectué un maximum d'itérations. Sachant que dans notre corpus certaines connexions sont très discriminantes (*retour* et *intermédiaire* par exemple), le réseau de neurones **apprend** en observant plusieurs exemples de ces connexions, que la probabilité qu'un vecteur contenant cette connexion appartienne à la même classe à laquelle ont appartenu les vecteurs précédents la contenant également, est élevée.

Dans notre travail, il existe deux façons d'utiliser NLC. Nous avons montré la première dans la section 2.2.3. La seconde manière est par appel API. Nous récupérons les identifiants de notre service bluemix, et nous utilisons les méthodes Watson pour créer un classifieur puis pour l'évaluer.
Nous obtenons des résultats corrects mais qui n'améliorent pas les règles.

Devant le manque de performance, il fallait trouver un moyen d'exploiter les réponses également.
Nous avons donc effectué la même séparation pour les réponses et les couples (questions ;réponses) que pour les questions seules, à savoir 75% pour l'apprentissage et 25% pour l'évaluation (*train* et *test* en anglais) en prenant soin de séparer exactement aux mêmes lignes que pour les questions sous peine de biaiser nos travaux.
Ensuite, comme que nous avions nettoyé le corpus d'apprentissage de questions, nous avons fait de même avec celui de réponses, et avec le couple de (questions ;réponses).

TABLE 4.2 – récapitulatif des sous-corpus avec et sans '0'

nom	*nbr mots '0'/≠'0'*	*f-mesure avec 0*	*sans 0*	*corpus de test*
question brut train	23 500/11 800	0.71	0.82	question brut test
question clean train	11 300/7 800	**0.74**	0.8	question brut test
reponse brut train	23 000/11 000	0.69	0.6	question brut test
reponse clean train	9 300/5 400	0.72	**0.88**	question brut test
question-reponse brut train	39 000 /21 500	0.72	0.78	question brut test
question-reponse clean train	31 000/17 300	0.68	0.87	question brut test
moyenne	23 000/12 500	0.71	0.79	N/D

Au total, nous possédions six sous-corpus d'apprentissage. Les corpus nettoyés sont appelés corpus **clean** et les corpus laissés tels quels **brut**.

Nous avons utilisé le même corpus d'évaluation pour tous les classifieurs **NLC** car ce qui nous intéresse dans le système c'est de classifier des questions. De plus, ces résultats nous servent de point de référence (*baseline* en anglais) pour le reste de nos expériences.
Ces résultats restent inférieur à ce que donne le système de règles pour l'ensemble du corpus de questions.
Nous avons tenté de comprendre pourquoi nous obtenions des résultats faibles.

Dans un premier temps, nous avons émis l'hypothèse que la forte proportion des exemples négatifs (plus de 50% du corpus) brouillait le réseau de neurones et faisait chuter la f-mesure.
Pour vérifier cette hypothèse, nous avons créé six autres sous corpus en enlevant tous les exemples négatifs (les '0') (tableau 4.2).

Nous avons construit douze sous-corpus :
Nous rappelons que nous possédons trois sous-corpus de 370 textes chacun. Celui de questions, de réponses, et de couples (question ;réponse). Nous les avons séparé entre corpus d'apprentissage et corpus de test aux mêmes endroits. Cependant, pour les tests, nous n'avons utilisé que le sous-corpus de test des questions brutes car ce sont les questions et uniquement les questions qui nous concernent telles qu'elles peuvent être posées par le end-user.
Ensuite, nous avons nettoyé les trois sous-corpus d'apprentissage pour alléger le contenu textuel du bruit possible et pour aider les classifieurs à s'entraîner. Nous possédions donc six sous-corpus.
Dans un dernier temps, pour vérifier l'hypothèse des exemples négatifs, nous les avons tous supprimés de ces six corpus. Pour ceux-là, nous avons également supprimé du corpus de test (les questions uniquement) les exemples négatifs. En définitive, nous avons créé douze sous-corpus d'entraînement et deux sous-corpus de test .

De manière générale (dans cinq cas sur six), le fait d'enlever les exemples négatifs améliorait grandement les scores.
Le problème demeurait là car nous ne pouvions pas faire sans eux, il fallait trouver un moyen de limiter leur impact négatif sur **NLC**.

Notre seconde hypothèse était que l'utilisation du taux de confiance pouvait améliorer les résultats du point de référence. L'inconvénient de cette solution restait la quantité de données perdues.

TABLE 4.3 – résultats avec les taux de confiance 0.8 et 0.9

nom	*résultats avec 0.8*	*avec 0.9*	*perte(en %) avec 0.8*	*avec 0.9*
question brut train	0.84	0.86	23	24
question clean train	**0.85**	0.87	22	32
reponse brut train	0.74	0.75	17	20
reponse clean train	0.83	**0.89**	18	27
q-r brut train	0.82	0.87	17	23
q-r clean train	0.78	0.86	22	27
moyenne	0.81	0.85	20	25

En effet, dans la mesure où la mise en place d'un seuil pour le taux de confiance induit que **NLC** ignore définitivement toutes les questions dont la confiance en la classe prédite ne dépasse pas ce seuil, nous perdons beaucoup de questions. Autrement dit, il ne faut pas que le nombre de questions ignorées soit trop élevé sinon notre système ne peut pas être considéré comme couvrant. Nous avons choisi comme seuils 0.8 et 0.9 de taux de confiance minimum (tableau 4.3)

racine et open-source

Nous avons estimé que le pourcentage de perte était trop élevé pour être utilisé (plus de 20% de perte). D'autre part, les autres seuils de taux de confiance que nous avons essayé ne permettaient pas d'obtenir des améliorations significatives sans une perte équivalente.

Toutefois, plusieurs autres facteurs pouvaient expliquer ces résultats.
Dans un premier temps, pour pouvoir évaluer correctement les classifieurs **NLC** et vérifier leur stabilité (le fait qu'il n'y ait pas de gros écarts de performance pour différents sous-ensembles du corpus), nous avons décidé de créer trois autres échantillons de corpus train/test.
Dans un deuxième temps, nous avons choisi de comparer les résultats de **NLC** avec une solution en libre-accès contenant plusieurs algorithmes, **scikit learn** disponible sur le langage informatique Python.
De plus, nous avons racinisé notre corpus(figure 4.4).
La **racinisation** est une méthode par laquelle nous ne conservons que la racine textuelle des mots. A l'inverse de la lemmatisation qui elle prend en compte la nature des mots et cherche à trouver le lemme de ces mots (le lemme de *chantons* est *chanter*, son infinitif ; le lemme de *chants* est *chant*, son substantif masculin singulier) et cherche une forme connue répondant au paradigme de ce mot, la racinisation, ou désuffixation (*stemming* en anglais), cherche à trouver le radical de ce mot en ignorant la validité orthographique de la forme finale racinisée (la racine de *chantons* sera *chant*, la racine de *chants* sera *chant*). L'inconvénient est que par exemple *je vais* et *nous allons* ne sera pas racinisé de la même façon : *je vai nous all*. Chaque méthode contient ses limites.

Cela permet de s'affranchir de la barrière prédicat-substantif et donc peut, dans certains cas, aider les algorithmes d'apprentissage automatique à corréler certaines phrases avec le même mot racinisé, ce qui n'était pas le cas dans la forme normale ou même lemmatisée.

```
"dan le cadr d'un stag un personnel militair peut-il utilis la voi routi civil san autoris d'utilis du véhicul personnel ",0
" un personnel militair va réalis un stag dan une écol militair à l'étranger. comment ser indemnis ce personnel ? ",3
" un personnel militair est en reconvers début janvi 2017 et souhait effectu un périod de reconvers de 6 mois au portugal. pouvez-vous me don la
" comment est indemnis un personnel civil (ouvri d'état ou fonctionnaire) effectu une périod de stag (indemn de stag ou frais de mission) ? ",3
" un personnel militair résid sur la commun de marignan et effectu son stag de reconvers à aix en provenc la distanc entre le deux local est d'en
" comment ser indemnis un personnel qui présent une attest de stag coch en héberg oner sur le lieu de stag alor que l'ordr de mission prévoit une
" l'attest de stag est-el obligatoir mêm si l'administr est héberg et nourr à titr gratuit ? ",0
" comment ser indemnis un personnel civil qui effectu une format à l'intérieur de sa résident administr ? ",3
" comment ser indemnis un personnel de se frais lorsqu le stag se termin le vendred en fin de matin : ij taux mission pour cet derni journ ? ",3
" comment doivent être présent le dossi pour lesquel la format se déroulent sur différent sit avec de condit d'héberg et de restaur différent ? c
" qu'entend-on par fermetur écol ? ",0
" comment sont indemnis le personnel effectu de aller retour quotidien dur leur périod de format ? ",3
```

FIGURE 4.4 – Extrait du corpus de questions racinisées

TABLE 4.4 – comparatif open-source et Watson(*forme normale*)

algorithme	*question clean*	*question brut*	*réponse clean*	*réponse brut*	*q-r clean*	*q-r brut*
SVM	0.76	0.78	0.72	0.66	0.78	0.74
Rég logistique	0.8	0.78	0.71	0.65	0.78	0.79
SVM linéaire	0.81	0.81	0.72	0.68	0.78	0.76
Arbre décision	0.5	0.55	0.65	0.57	0.57	0.68
Forêts	0.68	0.57	0.65	0.65	0.56	0.66
Perceptron	**0.84**	0.82	0.65	0.68	0.73	0.79
Bayésien naïf	0.71	0.72	0.64	0.65	0.7	0.77
AdaBoost	0.65	0.71	0.78	0.64	0.62	0.52
Watson	0.74	0.74	0.72	0.69	0.68	0.72

TABLE 4.5 – comparatif open-source et Watson(*forme racinisée*)

algorithme	*question clean*	*question brut*	*réponse clean*	*réponse brut*	*q-r clean*	*q-r brut*
SVM	0.77	0.76	0.74	0.63	**0.8**	0.74
Rég logistique	**0.8**	0.79	0.72	0.65	0.74	0.77
SVM linéaire	**0.8**	0.79	0.72	0.65	0.74	0.77
Arbre décision	0.48	0.55	0.59	0.54	0.57	0.68
Forêts	0.71	0.54	0.56	0.6	0.63	0.69
Perceptron	**0.8**	0.78	0.67	0.61	0.72	0.72
Bayésien naïf	0.72	0.77	0.63	0.67	0.76	0.71
AdaBoost	0.45	0.59	0.59	0.65	0.58	0.71
Watson	0.79	0.7	0.68	0.57	0.63	0.67

Enfin, comme il s'agit d'un pré-traitement automatique (utilisation de la bibliothèque Snowball en python), nous pouvons nous permettre de raciniser aussi le corpus d'évaluation.
Les tableaux 4.4 et 4.5 montrent un comparatif entre les algorithmes que nous avons utilisés dans **scikit-learn** et **Watson NLC**, l'un avec la forme normale, l'autre la forme racinisée du corpus.

D'une façon générale, nous pouvons observer que **NLC** ne surpasse pas les solutions de scikit-learn, quelle que soit la forme, notamment dans les réseaux de neurones (*Perceptron Multi Couches*). En outre, la racinisation ne permet pas d'améliorer les résultats. De ce fait, nous ne l'utilisons pas pour la partie finale des expériences.
Nous regardons dans le chapitre 5, les résultats de **NLC** et du meilleur algorithme de scikit learn (performance et rapidité) sur les trois autres échantillons. Car s'il est vrai que nous obtenons jusqu'à 0.83 de f-mesure dans un cas précis, nous ne pouvons pas considérer ce résultat comme significatif dans la mesure où nous ne savons pas encore quel sera-t-il sur une autre portion du corpus.

4.3 Conclusion

En définitive, ce qu'il faut retenir de ce chapitre c'est l'aspect visuel de notre corpus de départ, la façon dont nous l'avons manipulé, modifié, segmenté et enrichi, ainsi que les premiers résultats avec le système de règles et d'apprentissage automatique, avec plusieurs configurations différentes.
Les résultats restent assez décevants dans l'ensemble. Cependant, nous avons pu comprendre certains facteurs de chute des scores, comme la présence majoritaire d'exemples négatifs dans le corpus, ainsi que le faible taux de confiance général : pour 20% des questions, **NLC** n'est pas assez sûr de connaître la bonne réponse ; de fait il se trompe le plus souvent dans ces cas-là.
A travers ce chapitre nous avons tenté de répondre à un des éléments de notre problématique de départ : en effet, nos règles sont performantes tant que nous partons du principe que le end-user pose une question correctement. Dès lors que la question est mal posée, ou que sa réalisation linguistique nous est encore inconnue, elles ne fonctionnent plus. L'intérêt des méthodes statistiques est de ne pas tenir compte de ces considérations car elles s'entraînent à travers les exemples (induction). L'apprentissage automatique doit aider les règles lorsque celles-ci montrent leurs limites (qui sont celles de l'humain).
La question que pose ce chapitre c'est Comment faire pour améliorer les résultats des méthodes statistiques, que ce soit NLC ou les autres, et s'assurer d'une part que les résultats sont performants (au moins aussi bons que les règles) et fiables (réguliers quel que soit l'échantillon du corpus d'apprentissage et d'évaluation utilisés)? Nous apportons une tentative de réponse au chapitre suivant.

CHAPITRE 5

RÉSULTATS FINAUX

Sommaire

5.1 Introduction

Ce chapitre montre nos résultats définitifs sur les différents sous-corpus cités dans le chapitre 4, en comparant et en combinant les classifieurs entre eux, mais aussi avec l'approche symbolique. Dans une première section nous ne parlons que de l'apprentissage automatique, puis dans une seconde section nous abordons la partie finale de l'expérience : le système hybride.

5.2 Rivaux puis partenaires

En guise d'introduction à cette section, les tableaux 5.1 et 5.2 montrent un récapitulatif des résultats de nos différents classifieurs sur les quatre échantillons. En moyenne, le meilleur classifieur pour les différents échantillons reste faible par rapport aux règles. Les règles atteignent en moyenne 0.77 de f-mesure.

TABLE 5.1 – résultats des quatre échantillons avec **NLC** et les règles

classifieur	*test 1*	*test 2*	*test 3*	*test 4*
question clean	0.74	0.68	0.7	0.5
question brut	0.74	0.65	0.52	0.61
réponse clean	0.72	0.7	0.59	0.65
réponse brut	0.69	0.6	0.39	0.47
question-réponse clean	0.68	0.68	0.59	0.62
question-réponse brut	0.72	0.65	0.49	0.61
Watson Studio (règles)	0.75	**0.81**	0.81	0.72

TABLE 5.2 – résultats des quatre échantillons avec l'open source (régression logistique) et les règles

classifieur	*test 1*	*test 2*	*test 3*	*test 4*
question clean	0.78	**0.83**	0.74	0.57
question brut	0.78	0.78	0.72	0.65
réponse clean	0.71	0.66	0.68	0.62
réponse brut	0.65	0.66	0.51	0.63
question-réponse clean	0.78	0.75	0.74	0.71
question-réponse brut	0.79	0.74	0.7	0.55
Watson Studio (règles)	0.75	0.81	0.81	0.72

TABLE 5.3 – confrontation des classifieurs NLC

Soit une question q du corpus de test.
Soient les deux meilleurs classifieurs NLC C_1 et C_2
Soient θ et θ' les meilleurs taux de C_1 et C_2 et c_1 et c_2 les classes de ces θ
On renvoie la classification de c_1 ou c_2 selon le signe de $\theta - \theta$'

Tout d'abord, nous avons confronté les meilleurs classifieurs de **NLC** pour chaque classifieur, à savoir *question clean*, *question-réponse clean* et *réponse clean*.
Pour y parvenir, nous avons écrit un script qui utilise les trois classifieurs. Ainsi fait, nous utilisions le taux de confiance en les comparant entre eux, et nous fournissions la prédiction du meilleur d'entre eux (table 5.3) Cette technique nous a permis d'améliorer le système **NLC** en moyenne de 0.7 à 0.74 de f-mesure sur l'ensemble des quatre échantillons. Cependant, cela restait insuffisant si l'on voulait rendre le système compétitif.

Nous avons alors choisi, au lieu de mettre les classifieurs en rivalité, de les combiner en utilisant la technique du boosting [Freund, Y.,R. Schapire, 2001] adaptatif, appelé **AdaBoost** qui permet de combiner de façon itérative plusieurs classifieurs faibles afin de pondérer les erreurs du précédent. Il est implémenté dans scikit-learn.
Nous avons obtenu 0.76 de f-mesure. Cela améliorait notre point de référence mais restait en-deçà de ce dont nous avions besoin, c'est-à-dire améliorer les règles.

5.3 Le système hybride

Nous avons décidé de prendre les règles non pas comme notre point de référence absolu, mais comme un membre de ce système.
Dans la plupart des cas, les règles obtiennent la meilleure précision, par classe et par échantillon. Nous rappelons que la **précision** se mesure par le nombre d'éléments d'une classe correctement prédits sur le nombre total d'éléments de cette classe prédits. Pour les quatre échantillons la moyenne de précision est de 0.83.
A l'inverse, le **rappel** est plus élevé en exploitant les réseaux de neurones et la régression logistique. Le rappel est le nombre d'éléments d'une classe prédits par rapport au nombre total d'éléments de cette classe attendus.
Le rôle du système hybride consiste donc à mettre en valeur la précision des règles avec la puissance de rappel des approches par apprentissage automatique.(0.78 de rappel)

5.3.1 Par taux de confiance

Nous avons souhaité vérifier notre hypothèse de la précision en évaluant uniquement les règles. Dans la mesure où l'approche par règle ne choisit pas une classification mais renvoie toutes celles qu'elle a trouvées dans une question, nous avons vérifié si, parmi celles renvoyées se trouvait la classe attendue. Le résultat, sans surprise, était excellent : 0.91 de f-mesure. Cela signifiait que dans plus de neuf cas sur dix, les règles trouvent la bonne classe. Cela a été encore plus évident lorsque les règles trouvent au moins une seule classe, la f-mesure montait à 0.96. Autrement dit, si les règles trouvent au moins une classe, nous pouvons être presque certains qu'au moins une des classes trouvées est la bonne.
Malheureusement, notre algorithme naïf ne fonctionnait pas. Il fallait alors penser l'apprentissage automatique comme un juge devant identifier la bonne classe parmi celles renvoyées par les règles.

Ici, nous explorons la première tentative de système hybride. Nous regardions tout d'abord ce que proposent les règles, en utilisant l'algorithme que nous avons déjà cité :

si **repas-trajet**

sinon si **retour intermédiaire**

sinon si **principe indemnisation**

sinon...on regarde ce que propose **NLC**, à savoir le meilleur des trois classifieurs comme expliqué dans la section 5.2.

C'est un algorithme naïf que nous avons mis en place mais déjà, nous pouvions observer que, faute d'améliorer la classification par règles, l'approche hybride ne faisait pas trop baisser le score. De plus, la f-mesure se stabilisait à 0.76 mais ce qui est intéressant c'est que la précision et le rappel s'équilibraient. 0.78 pour la précision et 0.75 pour le rappel.
Cependant, cela restait moyen car nous ne nous contentions pas d'observer comment l'intégration de l'apprentissage automatique ne portait plus préjudice aux règles, il nous fallait améliorer le score des règles.
Cette fois, nous avons décidé de donner la priorité à l'apprentissage automatique. En espérant que les résultats changeraient et nous permettent de comprendre comment améliorer l'algorithme, nous avons effectué la réflexion inverse :

si, parmi les classifications des règles se trouvait le vainqueur de **NLC**

sinon si **repas-trajet** dans NLC,

sinon si **retour intermédiaire** dans NLC

sinon si **principe indemnisation** dans NLC

TABLE 5.4 – Système hybride basé sur le vote

échantillon	*règles*	*hybride NLC*	*hybride scikit-learn (régression logistique)*
échantillon 1	0.75	0.75	0.76
échantillon 2	0.81	0.81	0.84
échantillon 3	0.81	0.81	0.82
échantillon 4	0.72	0.73	0.75
moyenne	0.7725	0.775	0.7925

sinon..**0**

Les résultats, comme attendu, ont été mauvais, ce n'était pas une piste exploitable, les règles restaient meilleures que l'apprentissage automatique. Nous sommes revenus à notre première hypothèse, à savoir penser l'apprentissage automatique comme un juge devant identifier la bonne classe parmi celles renvoyées par les règles.

5.3.2 Par vote

C'est ainsi que nous avons imaginé une approche par vote. C'est une méthode simple qui se sert des six classifieurs. En fait, nous pouvons n'en utiliser que cinq.
En effet, comme nous nous trouvons face à une problématique de quatre classes (**repas**, **retour**, **remboursement**, et **hors-sujet**), nous avons pensé à faire voter les classifieurs, au lieu de les comparer ou de les combiner.
Telle une vraie élection démocratique, nous faisions face à x candidats (les quatre classes). Pour éviter une égalité, il nous fallait au minimum $x + 1$ électeurs. Nous avons alors sélectionné les cinq classifieurs les plus performants. Nous avons extrait la classe donnant le meilleur taux de confiance.
Nous les avons fait ensuite voter pour leur candidat.
Pour cela, nous avons écrit un script très simple, dans lequel nous stockions le candidat choisi par un classifieur-électeur et nous ajoutions une voix à chaque fois que ce même candidat était choisi par une autre classifieur. Ensuite, nous vérifiions que l'élu se trouvait parmi les classes renvoyées par les règles, sinon, nous appliquions notre premier algorithme naïf.
L'algorithme basé sur le vote demeure très simple, mais, peut-être pour cela, redoutablement efficace, car pour la première fois, nous avons amélioré les règles pour l'ensemble des échantillons (tableau 5.4).
En outre, le vote nous libérait du problème du seuil du taux de confiance. En effet, nous avions montré dans le chapitre 4 qu'un seuil de taux de confiance élevé permettait, moyennant perte de données, d'améliorer sensiblement les scores. Lors du vote, pour la plupart des questions d'évaluation, il existe au moins un classifieur dont le taux de confiance dépasse 0.9.

En somme, l'apprentissage automatique combiné aux règles, par l'algorithme basé sur le vote, parvient à améliorer les règles.

TABLE 5.5 – Système hybride basé sur les moyennes

soit i le nombre de classes c renvoyées par les règles ;
soit j le nombre de classifieurs C et θ leurs taux de confiance choisis ;
Pour chaque θ de θ_0 à θ_i dans C_0 à C_j :
On fait la moyenne de la somme des θ de chaque C pour chaque c (voir formule)
On renvoie la classe à la moyenne la plus élevée

Toutefois, il nous restait encore une hypothèse, celle non pas d'utiliser le gagnant du vote, mais de sélectionner le gagnant par élimination.

5.3.3 Par moyenne

En effet, si au lieu de valoriser le gagnant du vote, nous prenions en compte les règles à nouveau de façon prioritaire, et nous nous montrions plus souples à l'heure de choisir le vainqueur parmi les classifieurs de l'apprentissage automatique, nous pourrions améliorer les résultats.
Notre objectif était de regarder en prime abord toutes les classes que les règles avaient trouvées. Ensuite, selon celles, et seulement celles, qu'elles avaient trouvées, nous effectuions le vote. Cependant, comme **NLC** choisissait son propre vainqueur avec le taux de confiance, parfois, le vainqueur de **NLC** n'était pas présent parmi toutes les classes trouvées par les règles. C'est pourquoi le vote devenait truqué par cette configuration puisque certains électeurs se seraient retrouvés sans bulletin.

Nous avons trouvé une autre solution. Au lieu d'utiliser le vote, nous avons ignoré le seuil de confiance de **NLC**. Nous rappelons que **NLC** renvoie toutes les classes, dans l'ordre de leur taux de confiance. Nous avions comme habitude de ne considérer que la classe au plus haut taux de confiance, nous avons ignoré cet ordre.
Nous avons décidé de ne considérer que les classes présentes dans les règles. De cette façon, nous regardions parmi les classes trouvées par les règles, leur taux de confiance dans **NLC** pour les classifieurs choisis.
Contrairement au vote, où il nous en fallait au moins cinq, ici nous n'avons utilisé que les trois meilleurs sur l'ensemble des quatre échantillons. Nous avons additionné les taux de confiance pour chaque classe choisie et chaque classifieur choisi puis effectué une moyenne des totaux. Nous avons sélectionné la classe au résultat le plus élevé. De cette façon, nous étions certains que d'une part le vainqueur se trouvait parmi les classes choisies, d'autre part, nous nous affranchissions du vainqueur de **NLC**. Autrement dit :

$$\text{resultat} = avg(\sum_{k=0}^{j} \theta) \tag{5.1}$$

Cette méthode obtient des résultats contrastés. Si, avec la régression logistique nous parvenons à surpasser le système de vote, avec **NLC**, cela fait baisser le score (tableau 5.5)

TABLE 5.6 – Système hybride basé sur les moyennes

échantillon	*règles*	*hybride NLC*	*hybride scikit-learn (régression logistique)*
échantillon 1	0.75	0.8	0.84
échantillon 2	0.81	0.76	0.81
échantillon 3	0.81	0.68	0.75
échantillon 4	0.72	0.73	0.8
moyenne	0.7725	0.7425	0.8

TABLE 5.7 – récapitulatif des résultats

méthode	*score*
NLC (baseline)	0.71
règles	0.77
NLC + règles (vote)	0.775
Open-source + règles (vote)	0.79
NLC + règles (moyenne)	0.74
Open-source + règles (moyenne)	**0.8**

5.4 Conclusion

En définitive, ce qu'il faut retenir de ce chapitre c'est l'utilisation de plusieurs méthodes ayant pour but d'améliorer les résultats de départ, à la fois de l'apprentissage automatique, mais aussi de notre point de référence absolue, l'approche par règles. A travers deux méthodes différentes, une basée sur le vote, l'autre sur la moyenne des confiances, nous sommes parvenus à surpasser cette *baseline* et donc à démontrer qu'un système hybride, par la combinaison des règles et de l'apprentissage automatique est une bonne approche pour la classification textuelle multi-classes comme le justifient nos scores encourageants(table 5.6). Pour notre mémoire nous sommes parvenus à répondre à la problématique du système de question-réponse : Dans quelles mesures peut-on analyser correctement la question en entrée ?)

CHAPITRE 6

DISCUSSION

Sommaire

6.1 Introduction

Dans ce chapitre, nous présentons l'importance d'avoir amélioré nos résultats initiaux pour le système de question-réponse. D'autre part, nous montrons quelles sont les voies possibles d'amélioration de la classification, ainsi que les tâches à venir. Enfin, nous expliquons quelles sont les limites de ce système.

6.2 Impact du système hybride sur le système final

Comme nous l'avons décrit dans le chapitre 5, nous obtenons 0.8 de f-mesure pour notre système hybride de classification. Cette limite des 0.8 points était notre objectif. En effet, nous ne nous attendions pas à mieux compte tenu du peu de données en notre possession. D'autre part, la classification n'est pas la seule brique de notre système.
C'est là que le *chatbot* a un rôle à jouer : certes, si la thématique est mauvaise, le *chatbot* prend dès le départ un mauvais embranchement dans le dialogue. Pourtant, et ce n'est pas du tout négligeable, il demande à l'utilisateur des précisions sur plusieurs critères. S'il tombe sur des critères exclusifs à une certaine thématique différente de celle de départ, il corrige la thématique et transmet l'information au système de classification qui, pour ce type de question, saura quelle est la bonne thématique. Par exemple, *premier jour* / *dernier jour* sont des critères ne pouvant appartenir qu'à la thématique des **repas-trajet**. Cette thématique demeurant sous-représentée dans le corpus (5% des questions), il est fréquent qu'elle ne soit pas détectée. C'est celle qui a le plus bas taux de rappel en moyenne (0.35). Le critère ayant été récupéré (ou repéré) par le *chatbot*, le système hybride apprend

que c'est un critère d'ordre essentiel. D'autre part, le *chatbot* transmet deux autres informations : la nouvelle pondération de ce critère : si c'est un critère exclusif mais pas repéré comme tel par le système hybride, le *chatbot* corrige la pondération de ce critère et la transmet en retour au système. De plus, il parvient, aussi bien que faire se peut, à détecter le syntagme contenant ce critère et à le transférer comme tel au système hybride. Pour l'instant, la détection du syntagme reste très perfectible, mais à terme, elle permettra d'obtenir des améliorations significatives des résultats.

L'autre objectif du mémoire était de montrer l'importance de l'apprentissage automatique dans un système de classification par intentions. En effet, 77% de f-mesure restait un score assez encourageant et nous aurions pu en rester au système de règles, les résultats auraient été proches. Cependant, il ne faut pas oublier que nous avons trouvé au total dix-sept thématiques dans un corpus de dix mille questions. Autrement dit, écrire des règles pour autant de thématiques et de données prendrait un temps énorme, sachant que pour trois thématiques et quatre-cent questions nous avons mis plus d'un mois. Nous avons donc démontré que l'apprentissage automatique pouvait assister l'approche par règles voir l'améliorer et s'améliorer avec le temps sans intervention humaine, ce qui ne sera jamais le cas des règles.

L'état de l'art se situe aux alentours des 84% de f-mesure (cf TREC et SQuAD) pour les systèmes à domaine ouvert, et autour de 87% pour les systèmes à domaine fermé. Nos résultats n'atteignent pas l'état de l'art actuel mais notre méthode a donné des résultats encourageants. Dans les sections suivantes, nous parlons de certaines améliorations possibles pour nous rapprocher de l'état de l'art, ainsi qu'une synthèse sur les limites du système.

6.3 Améliorations possibles

6.3.1 Pré-traitement

Nous avons effectué une racinisation pour les données, sans succès. Nous avons également parlé de lemmatisation que nous n'avons pas mise en place (nous avions choisi de raciniser). Nous aurions pu effectuer une analyse morphologique. L'analyse morphologique, faite au niveau des règles par **Watson Studio** c'est le remplacement d'un mot par la catégorie grammaticale à laquelle il appartient (ex : *Stage supérieur à 4 semaines* devient *NOM ADJECTIF PREPOSITION ADJECTIF :NUMERAL NOM*). Nous ne l'avons pas exploitée pour l'apprentissage automatique.

D'autre part, si nous avions eu le temps, nous aurions pu mettre en place une campagne d'annotation pour l'ensemble du corpus afin de repérer non seulement les thématiques mais aussi tous les critères. Nous rappelons que le défaut des règles reste son faible rappel. Une annotation manuelle préalable aurait pu régler ce problème mais son coût reste très élevé.

TABLE 6.1 – Comparaison de résultats entre une méthode standard de classification et le co-apprentissage sur *question brut test*

algorithme	f-mesure
SVM	0,3
Watson	0,55
random tree	0,6
liblinear regression	0,61
iterative classification	0,62
régression logistique	0,67
co-apprentissage(perceptron)	**0,73**

6.3.2 Co-apprentissage

Le co-apprentissage est un algorithme très efficace dans deux cas de figure : lorsque nous ne possédons que peu de données déjà classifiées et nous voulons artificiellement augmenter la taille de notre corpus classifié (370 sur 10,000), et lorsque l'on peut diviser chaque texte de notre corpus en deux dimensions linguistiques distinctes mais appartenant au même univers. Nous avions fourni l'exemple des courriels mais dans notre cas de figure cela avait fonctionné pour les couples (questions ;réponses). Pourtant, à cause de la longueur du processus, (plus de 48h pour l'apprentissage) nous avions du arrêter les essais. Cela semblait prometteur : avec seulement 5% de corpus d'entraînement nous avions obtenu de meilleurs résultats d'évaluation qu'avec 75% de corpus d'entraînement. Cela était du au fait que le co-apprentissage avait ignoré nos classifications et avait interprété le corpus à sa façon (fonctionnement propre des réseaux de neurones artificiels) et que par conséquent notre classification humaine restait défectueuse. Autrement dit, le réseau de neurones artificiels "corrigeait" notre classification. Bien que nous n'ayons pas pu le mettre en place de façon fonctionnelle, le co-apprentissage nous avait permis de corriger des erreurs d'interprétation humaine dans certaines questions. C'est pour cela que nous pensions que ce serait une solution d'avenir, surtout avec la configuration actuelle de notre corpus (couple séparable en deux dimensions distinctes). C'est le co-apprentissage qui nous inspiré pour le système hybride : il nous est venu l'idée d'utiliser et de combiner plusieurs classifieurs. Dans le co-apprentissage, lorsque qu'un classifieur apprend, l'autre devient fonctionnel et alimente le premier classifieur avec sa prédiction, et vice-versa. D'autre part, une des méthodes d'évaluation du co-apprentissage est le système par vote où chaque classifieur se mue en électeur comme nous l'avons expliqué dans la section 5.3.2.

Nous avons évalué le co-apprentissage par rapport à d'autres algorithmes d'apprentissage automatique. Le tableau 6.1 montre son efficacité lorsque nous simulons un corpus de référence petit (5% du corpus total).

L'algorithme que nous avons implémenté est décrit dans le tableau 6.2

6.3.3 Watson Knowledge Studio

Nous l'avons cité en présentant **NLC**, Watson Knowledge studio, abrégé en **WKS** est un outil d'apprentissage automatique mais spécialisé dans le repérage d'entités nommées ou les mots semi-structurés comme les dates, les numéros de dossier ou les noms propres par exemple. Cela aurait pu nous être utiles pour repérer certains critères, en s'affranchissant des grammaires régulières spécifiques aux règles, surtout lorsque l'on s'aperçoit de la grande diversité linguistique dans notre corpus

TABLE 6.2 – Algorithme de co-apprentissage

Soit notre corpus C de 370 couples $(q; r)$. Nous séparons le corpus en trois sous-corpus :

Le corpus E de 20 couples étiquetés séparés en :
E_1 (les questions seulement) et E_2 (les réponses seulement).
Ces corpus servent à entraîner deux classifieurs Cl_1 et Cl_2
chacun sur une dimension de E.

Le corpus F de 260 couples non-étiquetés servant à alimenter artificiellement E.

Le corpus G composé de 80 questions qui sert à évaluer le système
lorsque les deux classifieurs sont entraînés.

Le but est de vider F dans E.

Tant qu'il reste des couples dans F

Pour i *de 0 à* $taille(F)$ *itérations*

on passe Cl_1 *en mode production avec* q *du* i_e *couple de* F

si le seuil de confiance θ *est jugé suffisant* :

on ajoute q *de* $F[i]$ *dans* $E2$
on ré-entraîne Cl_2

on supprime q *de* $F[i]$

on arrête

Pour j *de 0 à* $taille(F)$ *itérations* :

on teste la r *du* j_e *couple de* F *avec* Cl_2

si θ *est suffisant* :

on ajoute r *de* $F[j]$ *dans* E_1
on ré-entraîne Cl_1

on supprime r *de* $F[j]$

on arrête

on recommence

pour un même critère. Pour la suite du système, ce sera une piste importante de travail, nous pensons qu'exploiter cet outil est susceptible de nous aider à améliorer les scores finaux.

6.4 Limites du système

Ce système a été conçu pour un POC, donc par définition il n'a pas vocation à être commercialisé mais à convaincre le client que notre solution peut marcher et passer en production.
Le fait que nous n'ayons eu que six semaines pour construire le système, a limité nos options de rendre ce système robuste. En effet, les règles sont crées uniquement pour les trois thématiques du POC, ainsi que les corpus, les classifieurs et les configurations des algorithmes d'apprentissage automatique. De ce fait, il nous aurait été impossible de garantir que ce système fût opérationnel si nous ne l'avions pas essayé sur un autre corpus de question dans un autre registre. En effet, nous avons réuni un corpus de cinquante questions sur des problématiques de ressources humaines liées au remboursement des tickets de transport, des repas et des hébergements ainsi que des indemnités kilométriques lors de déplacements professionnels. Nous

obtenons de bons scores dans la détection des problématiques avec **NLC** et le *chatbot* **Watson Conversation**.
Notre système, sous réserve de certains changements notamment dans les règles, et dans le paramétrage de l'apprentissage automatique, a pu être réutilisé. Cependant, la faible quantité de données provoque un manque de stabilité dans le corpus (plus de 6 points d'écart d'un échantillon à l'autre). Ce problème peut toutefois être résolu avec plus de matière.

Conclusion générale

En définitive, ce qu'il faut retenir de ce mémoire c'est la tentative d'apporter des éléments de réponse à la problématique initiale : comment créer un système capable d'analyser et de répondre correctement à une question inconnue dans un domaine fermé, en sachant que cette question peut être imparfaite ou trop vague ? Les contraintes étaient doubles : mettre en œuvre ce projet dans des délais limités et utiliser uniquement la technologie IBM Watson dans toutes les étapes du système.

Pour y répondre, nous avons tout d'abord mis en place un système hybride de classification avec une approche par règles assistée par un système d'apprentissage automatique. La décision finale est prise à l'aide d'algorithmes basés sur le vote et sur des moyennes. Pour l'apprentissage automatique, nous avons utilisé plusieurs méthodes, en particulier des réseaux de neurones artificiels (Perceptron à couches multiples) et la régression logistique. Le résultat final arrive à 80% de f-mesure.

Pour compléter ce système, nous avons créé un agent conversationnel "chatbot" capable de récupérer des informations manquantes dans la question initiale et donc non détectés par le système de classification, mais aussi pour corriger les possibles mauvaises interprétations du système hybride et l'aider à s'améliorer. Cet agent fait en sorte que le processus d'analyse et de génération de réponse soit intégré au dialogue.

Le système reste limité par le faible volume des données initialement disponibles, et par la contrainte d'utiliser Watson qui ne nous a pas permis d'explorer des approches alternatives à celles disponible dans cet outil. La solution Watson étant par ailleurs peu fournie en terme d'exploration de contenu textuel, notre système à base de règles est resté superficiel, quand une analyse plus profonde aurait peut-être permis de préciser et d'améliorer les règles. En revanche, nous avons montré qu'il est possible de réaliser une classification supervisée avec peu de données et d'obtenir des résultats exploitables.

Lors d'un test chez le client notre système a bien répondu à suffisamment de question pour que le client semble satisfait. Il a été convaincu par la démonstration et souhaite poursuivre le travail dans les directions explorée, avant de peut-être passer en production et industrialiser notre solution.

BIBLIOGRAPHIE

[Tur,] – Cité page 17.

[Der,] Question analysis for a closed domain question answering system. – Non cité.

[Abdel Hady et al., 2009] Abdel Hady, M. F., Schwenker, F., and Palm, G. (2009). *Semisupervised Learning for Regression with Cotraining by Committee*, page 121130. Springer Berlin Heidelberg, Berlin, Heidelberg. – Non cité.

[Andor et al., 2016] Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally normalized transition-based neural networks. *CoRR*, abs1603.06042. – Non cité.

[Ayache et al., 2006] Ayache, C., Grau, B., and Vilnat, A. (2006). Equer the french evaluation campaign of questionanswering systems. In *LREC*. – Cité page 10.

[Baudis and Sedivy, 2015] Baudis, P. and Sedivy, J. (2015). *Modeling of the Question Answering Task in the YodaQA System*, page 222228. Springer International Publishing, Cham. – Cité page 15.

[Bauer and Berleant, 2012] Bauer, M. A. and Berleant, D. (2012). Usability survey of biomedical question answering systems. *Human genomics*, 6:17. – Non cité.

[Bearman, 2017] Bearman, A. L. (2017). Deep coattention networks for reading comprehension. – Cité page 16.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with cotraining. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92100, New York, NY, USA. ACM. – Non cité.

[Campillos Llanos et al., 2017] Campillos Llanos, L., Rosset, S., and Zweigenbaum, P. (2017). Automatic classification of doctorpatient questions for a virtual patient record query task. In *BioNLP 2017*, page 333341, Vancouver, Canada,. Association for Computational Linguistics. – Non cité.

[Cao et al., 2011] Cao, Y., Liu, F., Simpson, P., Antieau, L., Bennett, A., Cimino, J. J., Ely, J., and Yu, H. (2011). Askhermes: An online question answering system for complex clinical questions. *Journal of Biomedical Informatics*, 44(2):277 – 288. – Cité page 16.

[Clark and Manning, 2016] Clark, K. and Manning, C. D. (2016). Deep reinforcement learning for mentionranking coreference models. *CoRR*, abs/1609.08667. – Non cité.

[Cruchet et al., 2008] Cruchet, S., Gaudinat, A., and Boyer, C. (2008). Supervised approach to recognize question type in a qa system for health. 136:407–12. – Non cité.

[Derici et al., 2015] Derici, C., Çelik, K., Kutbay, E., Aydın, Y., Güngör, T., Özgür, A., and Kartal, G. (2015). *Question Analysis for a Closed Domain Question Answering System*, page 468482. Springer International Publishing, Cham. – Cité page 17.

[Dhingra et al., 2017] Dhingra, B., Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2017). Linguistic knowledge as memory for recurrent neural networks. *CoRR*, abs/1703.02620. – Non cité.

[Didaci et al., 2012] Didaci, L., Fumera, G., and Roli, F. (2012). *Analysis of Cotraining Algorithm with Very Small Training Sets*, page 719726. Springer Berlin Heidelberg, Berlin, Heidelberg. – Non cité.

[Dwivedi and Singh, 2013] Dwivedi, S. K. and Singh, V. (2013). Research and reviews in question answering system. *Procedia Technology*, 10(Supplement C):417 424. First International Conference on Computational Intelligence Modeling Techniques and Applications (CIMTA) 2013. – Non cité.

[Embarek, 2008] Embarek, M. (2008). *A question answering system in the medical domain the Esculape system*. Theses, Université ParisEst. – Cité page 16.

[EstivillCastro, 2002] EstivillCastro, V. (2002). Why so many clustering algorithms a position paper. *SIGKDD Explor. Newsl.*, 4(1):6575. – Cité page 56.

[Falco, 2014] Falco, M. (2014). *Answering multiple answer questions from the Web*. Theses, Universite Paris Sud Paris XI. – Non cité.

[Ferrucci, 2012] Ferrucci, D. (2012). Introduction to "this is watson". 56:11115. – Non cité.

[Ferrucci et al., 2010] Ferrucci, D., Brown, E., ChuCarroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. (2010). Building Watson an overview of the DeepQA project. *AI Magazine*, 31(3):5979. – Non cité.

[Ferrucci and Lally, 2004] Ferrucci, D. and Lally, A. (2004). UIMA An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3-4):327–348. – Cité page 11.

[Ferrucci et al., 2017] Ferrucci, D., Nyberg, E., Allan, J., Barker, K., Brown, E., ChuCarroll, J., Ciccolo, A., Duboue, P., Fan, J., Gondek, D., Hovy, E., Katz, B., Lally, A., Mccord, M., Morarescu, P., Murdock, B., Porter, B., Prager, J., Strzalkowski, T., and Zadrozny, W. (2017). Towards the open advancement of question answering systems. – Non cité.

[Foucault, 2013] Foucault, N. (2013). *Open domain questionanswering relevant document selection geared to the question*. Theses, Université Paris Sud Paris XI. – Non cité.

[Gliozzo et al., 2013] Gliozzo, A., Biran, O., Patwardhan, S., and McKeown, K. (2013). Semantic technologies in ibm watson. – Cité page 15.

[Gobeill et al., 2015] Gobeill, J., Gaudinat, A., Pasche, E., Vishnyakova, D., Gaudet, P., Bairoch, A., and Ruch, P. (2015). Deep question answering for protein annotation. 2015. – Cité page 16.

[Grau et al., 2006] Grau, B., Ligozat, A., Robba, I., Vilnat, A., and Monceaux, L. (2006). Frasques a question answering system in the equer evaluation campaign. In *LREC*. – Non cité.

[Hammo et al., 2002] Hammo, B., Abu-Salem, H., and Lytinen, S. (2002). Qarab a question answering system to support the arabic language. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, SEMITIC '02, pages 1–11, Stroudsburg, PA, USA. Association for Computational Linguistics. – Cité page 17.

[Harabagiu et al., 2000] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., and Morarescu, P. (2000). Falcon boosting knowledge for answer engines. page 479488. – Cité page 15.

[HIRSCHMAN and GAIZAUSKAS, 2001] HIRSCHMAN, L. and GAIZAUSKAS, R. (2001). Natural language question answering the view from here. *Natural Language Engineering*, 7(4):275300. – Non cité.

[Hou and Tsai, 2014] Hou, W. and Tsai, B. (2014). *An Answer Validation Concept Based Approach for Question Answering in Biomedical Domain*, page 148159. Springer International Publishing, Cham. – Non cité.

[Hsu et al., 1995] Hsu, F., Campbell, M. S., and Hoane, Jr., A. J. (1995). Deep blue system overview. In *Proceedings of the 9th International Conference on Supercomputing*, ICS '95, page 240244, New York, NY, USA. ACM. – Non cité.

[Ittycheriah and Roukos, 2001] Ittycheriah, A. and Roukos, S. (2001). Ibm's statistical question answering system trec11. – Non cité.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering a review. *ACM Comput. Surv.*, 31(3):264323. – Cité page 56.

[Kalyanpur et al., 2011] Kalyanpur, A., Patwardhan, S., Boguraev, B., Lally, A., and ChuCarroll, J. (2011). Factbased question decomposition for candidate answer reranking. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 20452048, New York, NY, USA. ACM. – Non cité.

[Katz et al., 2006] Katz, B., C. Borchardt, G., and Felshin, S. (2006). Natural language annotations for question answering. – Cité page 10.

[Kaufmann et al., 2006] Kaufmann, E., Bernstein, A., and Zumstein, R. (2006). Querix a natural language interface to query ontologies based on clarification dialogs. – Cité page 15.

[Kiritchenko and Matwin, 2001] Kiritchenko, S. and Matwin, S. (2001). Email classification with cotraining. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, CASCON '01, page 8. IBM Press. – Non cité.

[Kolomiyets and Moens, 2011] Kolomiyets, O. and Moens, M. (2011). A survey on question answering technology from an information retrieval perspective. 181:54125434. – Cité page 15.

[Kumar et al., 2015] Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2015). Ask me anything dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285. – Cité page 16.

[Lehnert, 1977] Lehnert, W. G. (1977). A conceptual theory of question answering. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence Volume 1*, IJCAI'77, page 158164, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. – Non cité.

[Litkowski, 1999] Litkowski, K. (1999). Question-answering using semantic relation triples. – Cité page 16.

[Llanos et al., 2015] Llanos, L. C., Bouamor, D., Bilinski, E., Ligozat, A., Zweigenbaum, P., and Rosset, S. (2015). Description of the patientgenesys dialogue system. In *SIGDIAL Conference*. – Non cité.

[Luque, 2006] Luque, J. (2006). Geovaqa: a voice activated geographical question answering system. – Cité page 17.

[Martin et al., 2001] Martin, A. I., Franz, M., and Roukos, S. (2001). Ibm's statistical question answering systemtrec10. In *In Proceedings of the Tenth Text REtrieval Conference (TREC*. – Cité page 10.

[Mishra and Jain, 2016] Mishra, A. and Jain, S. K. (2016). A survey on question answering systems with classification. *Journal of King Saud University Computer and Information Sciences*, 28(3):345 361. – Non cité.

[Neves et al., 2017] Neves, M. L., Eckert, F., Folkerts, H., and Uflacker, M. (2017). Assessing the performance of olelo, a real-time biomedical question answering application. In *BioNLP*. – Cité page 16.

[Nguyen et al., 2009] Nguyen, D. Q., Nguyen, D. Q., and Pham, S. B. (2009). A vietnamese question answering system. In *2009 International Conference on Knowledge and Systems Engineering*, page 2632. – Cité page 15.

[Nigam and Ghani, 2000a] Nigam, K. and Ghani, R. (2000a). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 86–93, New York, NY, USA. ACM. – Cité page 48.

[Nigam and Ghani, 2000b] Nigam, K. and Ghani, R. (2000b). Understanding the behavior of cotraining. – Non cité.

[Nouvel,] Nouvel, D. Classification automatique. – Non cité.

[Nyberg et al., 2002] Nyberg, E., Mitamura, T., Carbonell, J., P. Callan, J., Collins-Thompson, K., Czuba, K., Duggan, M., Hiyakumoto, L., Hu, N., Huang, Y., Ko, J., Lita, L., Murtagh, S., Pedro, V., and Svoboda, D. (2002). The javelin question-answering system at trec 2002. – Cité page 16.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120. – Cité page 16.

[Pan et al., 2017] Pan, B., Li, H., Zhao, Z., Cao, B., Cai, D., and He, X. (2017). Memen multi-layer embedding with memory networks for machine comprehension. – Cité page 16.

[Pedregosa, 2011] Pedregosa (2011). Scikitlearn machine learning in python. – Cité page 48.

[Punyakanok et al., 2004] Punyakanok, V., Roth, D., and Yih, W. (2004). Mapping dependencies trees an application to question answering. – Cité page 16.

[Rajpurkar et al., 2016] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Questions for machine comprehension of text. *CoRR*, abs160605250. – Cité page 16.

[Rakotomalala, 2011] Rakotomalala, R. (2011). Pratique de la rÃ©gression logistique. – Non cité.

[Rao et al., 2016] Rao, J., He, H., and Lin, J. (2016). Noisecontrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 19131916, New York, NY, USA. ACM. – Non cité.

[Saha and Ekbal, 2013] Saha, S. and Ekbal, A. (2013). Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition. *Data & Knowledge Engineering*, 85(Supplement C):15 39. Natural Language for Information Systems Communicating with Anything, Anywhere in Natural Language. – Non cité.

[Shao, 2017] Shao, L. (2017). Generating long and diverse responses with neural conversation models. *CoRR*, abs/1701.03185. – Cité page 17.

[Shao et al., 2017] Shao, L., Gouws, S., Britz, D., Goldie, A., Strope, B., and Kurzweil, R. (2017). Generating long and diverse responses with neural conversation models. *CoRR*, abs/1701.03185. – Non cité.

[Tax et al., 2001] Tax, D., van Breukelen, M., Duin, R., and Kittler, J. (2001). Combining multiple classifiers by averaging or by multiplying? 33:14751485. – Non cité.

[Vinyals and Le, 2015] Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *CoRR*, abs/1506.05869. – Cité page 17.

[Waltz, 1978] Waltz, D. L. (1978). An english language question answering system for a large relational database. *Commun. ACM*, 21(7):526539. – Non cité.

[Wan, 2009] Wan, X. (2009). Co-training for cross-lingual sentiment classification. 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-Studio Volume 1 - Volume 1*, ACL '09, pages 235–243, Stroudsburg, PA, USA. Association for Computational Linguistics. – Cité page 48.

[Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):3645. – Cité page 17.

[Welty et al., 2012] Welty, C., Murdock, J. W., Kalyanpur, A., and Fan, J. (2012). *A Comparison of Hard Filters and Soft Evidence for Answer Typing in Watson*, page 243256. Springer Berlin Heidelberg, Berlin, Heidelberg. – Non cité.

[Wiseman et al., 2016] Wiseman, S., Rush, A. M., and Shieber, S. M. (2016). Learning global features for coreference resolution. *CoRR*, abs/1604.03035. – Cité page 17.

[Wozniak, 2014] Wozniak, M. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16. – Non cité.

[Xiong et al., 2016] Xiong, C., Zhong, V., and Socher, R. (2016). Dynamic coattention networks for question answering. *CoRR*, abs1611.01604. – Cité page 16.

[Yao et al., 2014] Yao, X., Berant, J., and Van Durme, B. (2014). Freebase qa information extraction or semantic parsing? – Cité page 15.

[Yih et al., 2014] Yih, S. W.-t., He, X., and Meek, C. (2014). Semantic parsing for single-relation question answering. – Cité page 15.

[Yih et al., 2015] Yih, W., Chang, M., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 Long Papers)*, page 13211331, Beijing, China. Association for Computational Linguistics. – Cité page 15.

[Yu et al., 2007] Yu, H., Lee, M., Kaufman, D., Ely, J., Osheroff, J. A., Hripcsak, G., and Cimino, J. (2007). Development, implementation, and a cognitive evaluation of a definitional question answering system for physicians. *Journal of Biomedical Informatics*, 40(3):236 – 251. – Cité page 16.

[Zheng, 2002] Zheng, Z. (2002). Answerbus question answering system. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, page 399404, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. – Cité page 16.

ANNEXE A

EVALUATIONS

A.1 Formules mathématiques

Le rappel (formule A.1) mesure le nombre d'éléments correctement étiquetés par le système (vrais positifs) rapporté au nombre d'éléments étiquetés dans la référence (vrais positifs et faux négatifs).

$$\text{Rappel} = \frac{\text{vrais positifs}}{\text{vrais positifs + faux négatifs}} \tag{A.1}$$

La précision (formule A.2) mesure le nombre d'éléments correctement étiquetés par le système (vrais positifs) rapporté au nombre total d'éléments étiquetés par le système (vrais et faux positifs).

$$\text{Précision} = \frac{\text{vrais positifs}}{\text{vrais positifs + faux positifs}} \tag{A.2}$$

La F-mesure (formule A.3) est la moyenne harmonique pondérée du rappel et de la précision. La valeur accordée à β permet de pondérer le rappel ou la précision, ou d'équilibrer les deux mesures (avec $\beta = 1$).

$$\text{F-mesure} = \frac{(1+\beta^2) \times \text{ précision } \times \text{ rappel}}{\beta^2 \times \text{ précision } + \text{ rappel}} \tag{A.3}$$

A.2 résultats

	rule_based		Nous utilisons la f-mesure pour évaluer NLC							
	0,75									
		supervised (75/25)					proportion(%)			
	avec 0 brut	sans 0 brut	avec 0 clea	sans 0 clea	average		0,8	0,9	0,95	average
questions	0,71	0,82	**0,74**	0,8	0,7675	question br	77	76	67	73,3333333
reponses	0,69	0,6	0,72	0,88	0,7225	question cl	78	68	65	70,3333333
questions+	0,72	0,78	0,68	0,87	0,7625	reponse br	**83**	**80**	**76**	79,6666667
average	0,70666667	0,73333333	0,71333333	0,85	0,75083333	reponse cle	82	73	61	72
						qr brut	**83**	77	68	76
		supervised(taux-confiance >= 0,8)				qr clean	78	73	66	72,3333333
						average	80,1666667	74,5	67,1666667	
	avec 0 brut	sans 0 brut	avec 0 clea	sans 0 clean						
questions	0,84	0,95	**0,85**	0,82	0,865					
reponses	0,74	0,7	0,83	0,91	0,795					
questions+	0,82	0,78	0,78	0,87	0,8125					
average	0,8	0,81	0,82	0,86666667	0,82416667					
		supervised(>= 0,9)								
	avec 0 brut	sans 0 brut	avec 0 clea	sans 0 clean						
questions	0,86	1	0,87	0,85	0,895					
reponses	0,75	0,68	**0,89**	0,91	0,8075					
questions+	0,87	0,8	0,86	0,88	0,8525					
average	0,82666667	0,82666667	0,87333333	0,88	0,85166667					

Analyse : . il y a une grosse différence de performance dès que l'on place un taux de confiance au dessus de 80% (+ 7 points en moyenne)
. il n'y a pas beaucoup de différence entre 0,8 et 0,9 de taux de confiance
. Les repas trajets sont la classe la plus défaillante (0,1 de F-1 score en moyenne)

	questions t	10 90	Co-training
svm	0,3		
Watson	0,55		
random tre	0,6		
liblinear	0,61		
iterative cl	0,62		
logistic reg	0,67		

FIGURE A.1 – premiers résultats avec NLC et le co-apprentissage

benchmark adaboost							
	qu clean	qu brut	r clean	r brut	qr clean	qr brut	
sample 2	0,66	**0,71**	0,7	0,64	0,65	0,53	0,64833333
sample 3	0,52	0,44	0,5	0,45	**0,53**	0,45	0,48166667
sample4	**0,66**	0,31	0,48	0,49	0,28	0,3	0,42
average	0,61333333	0,48666667	0,56	0,52666667	0,48666667	0,42666667	0,51666667

benchmark SVM							
	qu clean	qu brut	r clean	r brut	qr clean	qr brut	
sample 2	0,77	0,77	0,72	0,66	**0,78**	0,74	0,74
sample 3	**0,74**	0,67	0,65	0,4	0,73	0,66	0,64166667
sample4	0,59	0,67	**0,7**	0,62	0,64	0,58	0,63333333
average	0,7	0,70333333	0,69	0,56	0,71666667	0,66	0,67166667

benchmark régression logistique							
	qu clean	qu brut	r clean	r brut	qr clean	qr brut	
sample 2	0,77	**0,81**	0,69	0,67	**0,81**	0,77	0,75333333
sample 3	**0,75**	0,69	0,7	0,51	0,73	0,72	0,68333333
sample4	0,59	0,65	0,66	0,59	**0,69**	0,55	0,62166667
sample 5	0,75	0,71	0,67	0,64	**0,76**	0,79	0,72
average	0,715	0,715	0,68	0,6025	0,7475	0,7075	0,69458333

FIGURE A.2 – benchmark sur les meilleurs algorithmes utilisés sur les 4 échantillons

Index