Améliorer un agent conversationnel : prendre en compte à la volée des retours utilisateurs

Maxime Arens

Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles
Lille, France, 28 juin au 2 juillet 2021

Améliorer un agent conversationnel : prendre en compte à la volée des retours utilisateurs

Maxime Arens[1,2]
(1) IRIT, Cours Rose Dieng-Kuntz, 31400 Toulouse, France
(2) Synapse Développement, 7 Boulevard de la Gare, 31500 Toulouse, France
maxime.arens@irit.fr

RÉSUMÉ

Nous présentons une approche améliorant la pertinence des réponses d'un système conversationnel de question-réponse en profitant de l'expérience passée du système. Un agent conversationnel déployé au contact d'utilisateurs peut en effet profiter de retours afin d'améliorer la validité de ces futures réponses. Les systèmes de question-réponse fonctionnent généralement autour d'un modèle rapprochant sémantiquement une question à une ou plusieurs réponses potentielles. Ici, nous prenons en compte le cas où le modèle de correspondance rapproche une question à une liste de réponses associées à un score de pertinence. Une approche classique pour prendre en compte les retours d'utilisateurs, est de les utiliser pour augmenter le nombre de données de réentrainement du modèle de rapprochement sémantique. Nous proposons une approche différente, impactant le score des réponses potentielles, où nous prenons en compte « à la volée » les retours utilisateurs : entre le moment où l'utilisateur pose une nouvelle question et celui où le système lui répond.

ABSTRACT

Improve a conversational agent : considering on the fly user feedback.

We present an approach to improve the relevance of a conversational question answering system by leveraging previous user feedback. A dialog system deployed in contact of users can take into accounts feedbacks to improve the relevance of its answers. Question answering systems usually work through models matching a question with one or multiple answers. Here we consider the case where the model matches a question to a list of answers scored by relevance. A classical approach of considering user feedback is to augment the training data used to retrain the matching model. Here we suggest a different approach, impacting answers scores, by considering "on the fly" the feedbacks : between when the user asks a new question and when the system responds.

MOTS-CLÉS : Question-réponse conversationnelle ; Retours utilisateurs ; Similarité entre questions ; Apprentissage actif.

KEYWORDS: Conversational question answering ; User feedback ; Question similarity ; Active Learning.

Volume 2 : 23e REncontres jeunes Chercheurs en Informatique pour le TAL (RECITAL), pages 2–14.

1 Introduction

Les systèmes conversationnels de question-réponse permettent, au fil d'une conversation, suite à une question formulée sous la forme d'une requête en langage naturel, de retourner une réponse issue d'une base de connaissances (Reddy *et al.*, 2019). De telles réponses peuvent être générées à partir d'informations ou bien, comme dans le cas nous concernant, extraites d'un corpus de document (Hoi *et al.*, 2018). Ces agents mêlent à la fois des techniques issues de la discipline du Traitement Automatique des Langues (TAL) et de celle de la Recherche d'Information (RI) (Belkin *et al.*, 1995). En étudiant le fonctionnement de ces systèmes d'un point de vue chronologique, la compréhension de la requête utilisateur (Qu *et al.*, 2019) est plus précisément une tâche de Compréhension du Langage Naturel (sous-branche du TAL), tandis que l'identification du document contenant la réponse et son extraction (Tellex *et al.*, 2003) appartiennent plus au domaine de la RI.

Certains agents évoluent sur des domaines très ouverts et grand public (Rajpurkar *et al.*, 2016; Qu *et al.*, 2020) tandis que d'autres se focalisent sur des domaines restreints et techniques (Campos *et al.*, 2020b). La démocratisation de ces systèmes au sein des entreprises, en tant qu'outil de support de la relation client ou bien à des fins internes de gestion de ressources informatives (Gao *et al.*, 2019), rend l'adaptation de l'agent au domaine de l'entreprise souvent nécessaire. Cette spécialisation, nécessaire pour élaborer des systèmes conversationnels (Aliannejadi *et al.*, 2019) portant sur des sujets précis et à haute technicité, nécessite des données d'entraînement (Campos *et al.*, 2020b). Ces données d'entraînement sont souvent des données annotées manuellement par des experts (par exemple par l'annotation d'une réponse correcte pour une certaine réponse). Or, le recours à des experts, particulièrement sur des domaines spécialisés et techniques, est onéreuse.

Un véritable enjeu existe donc pour limiter le recours à ces experts afin de réduire les ressources nécessaires à l'adaptation de ces systèmes. Une façon de répondre à cet enjeu est de déployer un système conversationnel partiellement spécialisé, de le faire interagir avec des utilisateurs puis de prendre en compte leurs retours pour améliorer la performance de l'agent (Hancock *et al.*, 2019). Les retours utilisateurs peuvent évaluer chaque tour de la conversation individuellement ou donner une appréciation globale de la conversation. Nous utilisons le cas où l'utilisateur évalue des tours de conversation, plus précisément, évalue binairement (positivement ou négativement) la réponse du système à sa question.

Puisque ici le système conversationnel repose sur une interaction avec des agents humains (Li *et al.*, 2016) pour son entraînement, il est important que les mécanismes d'apprentissage du système prennent en compte certaines particularités de ce cas d'utilisation réelle. Tout d'abord, les requêtes d'utilisateurs humains, bien que véhiculant parfois le même questionnement, sont souvent formulées de manières différentes et peuvent contenir des fautes (Christmann *et al.*, 2019). Ensuite, les retours des utilisateurs étant par définition subjectifs, des désaccords peuvent naître autour de la perception de la qualité d'une réponse apportée par le système. De plus, les utilisateurs peuvent volontairement ou involontairement donner des retours négatifs sur une réponse qu'un expert jugerait correcte. Enfin, le niveau d'amélioration du système est dépendant du nombre, de la qualité et de la portée des retours utilisateurs.

Une approche classique pour améliorer un système conversationnel à partir de retours utilisateurs, est de se servir de ces retours comme données supplémentaires lors d'un réentrainement du modèle rapprochant une question à des réponses (Campos *et al.*, 2020a). Cette approche nécessite tout d'abord que le module de rapprochement sémantique question-réponse soit réentrainable et qu'une boucle de réentrainement/redéploiement du module soit implémentée (Liu *et al.*, 2018). Nous présentons

ici une approche différente permettant de s'affranchir de ces deux conditions. On suppose que le module de rapprochement sémantique retourne une liste de réponses ayant chacune un score de pertinence. Tout d'abord, au moment où l'utilisateur pose sa question, nous récupérons grâce à un modèle d'équivalence entre questions le plus grand nombre de retours utilisateurs liés à des réponses apportées par le système pour des questions équivalentes (Prabowo & Budi Herwanto, 2019). Nous formons alors des quadrets comprenant chacun une question équivalente à la question source, une réponse apportée par le système et les retours utilisateurs binaires sur ce couple question-réponse. Grâce à une fonction prenant en entrée : le score originel des réponses envisagées pour la question et les retours d'utilisateurs évaluant ces réponses par rapport à des variantes passées de sa question, l'algorithme que nous présentons calcule un nouveau score pour chacune des réponses potentielles. Enfin, le système retourne à l'utilisateur la réponse ayant le meilleur score de pertinence, après ajustement du score de pertinence au moyen de la prise en compte de ses expériences passées.

2 Approche proposée

Dans cette section nous allons détailler l'approche que nous proposons dans cet article. Nous commencerons par expliciter l'ensemble du processus d'un point de vue général. Ensuite, nous aborderons en détail notre choix de modèle d'équivalence et son fonctionnement. Enfin, nous discuterons de la fonction modifiant le score des réponses en fonction des retours utilisateurs.

2.1 Architecture

Lorsqu'un utilisateur pose une question, le système obtient une liste de réponses potentielles et retourne à l'utilisateur, celle ayant le score de pertinence le plus élevé. Suite à cet échange, l'utilisateur peut ensuite évaluer la réponse du système en la jugeant satisfaisante ou non satisfaisante. Ce retour est matérialisé par une mise en mémoire en base de données du quadret (question utilisateur ; réponse du système ; nombre de retours utilisateurs positifs ; nombre de retours utilisateurs négatifs). La question utilisateur est donc la requête faite par l'utilisateur en langage naturel. La réponse du système est un extrait d'un document faisant partie de la base de connaissances de l'agent conversationnel. Finalement, le retour utilisateur est une valeur binaire : 0 pour une réponse n'ayant pas satisfait l'utilisateur, 1 pour une réponse l'ayant satisfait. On note que de par le fonctionnement des bases de données, deux questions utilisateurs non identiques (avec une faute d'orthographe par exemple) constitueront deux lignes différentes dans la table de données, leurs nombres de retours utilisateurs ne seront donc pas rassemblés dans la base.

L'approche que nous proposons consiste en l'ajout d'un processus entre le moment où l'utilisateur pose sa question et le système lui répond. Ce processus est constitué de plusieurs étapes. Pour commencer, nous récupérons la liste des réponses envisagées par le système suite à la question de l'utilisateur. Pour chacune de ces réponses, nous collectons en base de données la liste des quadrets contenant les questions utilisateurs ayant fait remonter cette réponse ainsi que les retours faits par les utilisateurs sur leur satisfaction liée à cette réponse pour leur question. À l'aide du modèle d'équivalence entre questions, le système mis en oeuvre compare ensuite la nouvelle question utilisateur avec chacune des questions utilisateurs contenues dans les listes de quadrets. Le système identifie donc quels sont les quadrets contenant des retours utilisateurs sur la pertinence des réponses à des variantes de la question posée par l'utilisateur. Pour chacune des réponses contenues dans la

liste des réponses envisagées nous avons donc potentiellement une liste de quadrets contenant de l'information pertinente. Le système itère alors sur la liste de réponses potentielles afin de modifier le score de pertinence de cette réponse grâce aux nouvelles informations pertinentes obtenues. Cette fonction de modification prend le score de pertinence initial, les sommes des retours utilisateurs positifs et négatifs associés à cette réponse, et retourne un nouveau score de pertinence. Nous trions de nouveau la liste de réponse pour prendre en compte les potentiels changements de classement entre les réponses. Enfin, nous renvoyons à l'utilisateur la réponse en haut du classement, celle qui a le plus haut score de pertinence.

2.2 Similarité entre questions

Afin d'expliquer le fonctionnement du module d'équivalence entre questions, nous commencerons par discuter de ces entrées et des ces sorties. En entrée, nous avons la question posée par l'utilisateur et une liste de questions utilisateurs (le même ou d'autres utilisateurs) posées par le passé au système. En sortie, nous avons les indices des questions utilisateurs passées, considérées comme étant des variantes de la question que vient de poser l'utilisateur. Nous entendons par variante, une question ayant le même sens, contenant des fautes d'orthographe ou étant une reformulation de la question posée par l'utilisateur.

Cette détection d'équivalence de questions est une étape critique du processus proposé. En effet, étant située entre le moment où l'utilisateur pose sa question et obtient sa réponse, cette détection ne peut pas durer plus d'un certain temps si on ne veut pas impacter le ressenti de l'utilisateur. De plus, la précision du modèle est un critère très important. Dans le cas où, le modèle se tromperait en classifiant une question comme équivalente, le système global prendrait alors en compte des retours d'utilisateurs qualifiant un couple question-réponse n'ayant rien à voir avec celui qu'il essaie d'évaluer.

Avec ces deux idées en tête, nous avons entraîné un classifieur permettant d'identifier une question comme variante d'une autre. Nous sommes partis sur la piste de prendre des modèles évaluant la similarité entre deux phrases (Agirre *et al.*, 2012), puis de spécialiser ces modèles sur la tâche nous intéressant. Premièrement nous avons constitué un corpus de données pour l'entraînement et l'évaluation de tels modèles. Afin d'être au plus proche de notre cas d'utilisation réelle, nous avons récupéré l'ensemble des requêtes utilisateurs faites à un agent conversationnel lors de son déploiement. Cet agent est destiné au grand public sur le domaine de l'entrepreneuriat en France. Nous avons combiné entre elles les différentes requêtes utilisateurs afin de former des couples questions utilisateurs. À l'aide d'un modèle de similarité entre phrases, nous avons ensuite formé un ensemble de données plus petit composé de couples de questions potentiellement similaires. Le corpus, ainsi obtenu, atteint un peu plus de 4 000 couples de questions. Deux experts ont alors annotés à la main chacun de ces couples comme étant ou non des variantes d'une même question. Grâce à une étape de réconciliation, les experts se sont accordés sur le label de chaque couple.

Une fois le corpus obtenu, l'étape suivante a été de réfléchir sur les caractéristiques des questions sur lesquelles nos modèles allaient s'appuyer pour faire leur classification. En effet, ces modèles fonctionnent en calculant la distance [1] entre les représentations distribuées (plongements lexicaux) des mots composants les questions (Kusner *et al.*, 2015). Le temps d'exécution du calcul des caractéristiques des questions pouvant être trop lent pour les applicatifs visés, nous avons essayé un

1. Le calcul des distances mentionnées dans la Table 1 est réalisé entre la moyenne des vecteurs représentatifs des mots de chacune des questions.

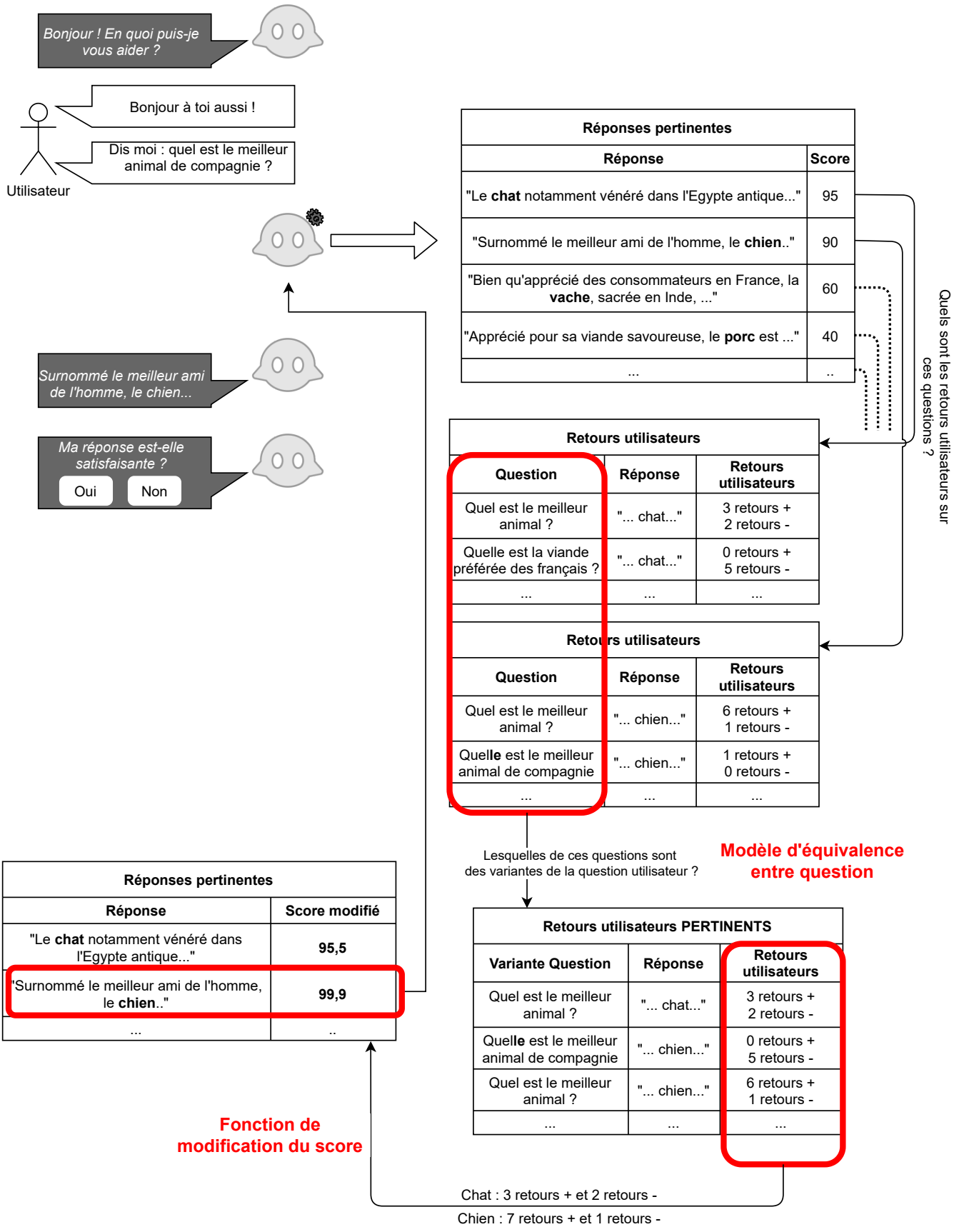

FIGURE 1 – Exemple d'exécution de l'algorithme proposé

important spectre de caractéristiques : certaines simples à calculer et d'autres un peu plus complexes à obtenir.

Caractéristiques simples	Caractéristiques intermédiaires	Caractéristiques complexes
Nombre de mots en commun	Taille de la plus grande sous-phrase en commun	Distance cosinus
Nombre de mots total	Pourcentage de tokens ayant du sens en commun	Distance de Manhattan
Pourcentage de mots en commun	Pourcentage de tokens vide de sens en commun	Distance de Canberra
Premier mot identique	Distance de Levenshtein	Distance euclidienne
Dernier mot identique		Distance de Minkowski

TABLE 1 – Tableau des caractéristiques essayées

Nous avons ensuite évalué différents classifieurs, avec certaines des caractéristiques de la Table 1 afin de trouver celui combinant les meilleures performances en matière de f1-score et de temps d'exécution. Les algorithmes derrière ces classifieurs reposent sur la minimisation de la différence entre les différentes valeurs caractéristiques des questions listées dans la Table 1.

Noms des méthodes
Recherche des plus proches voisins
Gradient stochastique (Bottou, 2010)
Forêt d'arbres décisionnels
Régression logistique
Machines à vecteurs de support
XGBoost (Chen & Guestrin, 2016)

TABLE 2 – Tableau des méthodes de classification évaluées

Suite aux évaluations présentées dans la prochaine section de cet article, nous avons donc choisi un classifieur reposant sur l'algorithme XGBoost répondant à nos critères de précision et de performance temporelle.

2.3 Modification du score

Une grande partie de l'efficacité de l'approche proposée dans cet article est liée à la fonction modifiant le score de pertinence des réponses potentielles en fonction des retours utilisateurs passés. Cette fonction doit répondre à certaines exigences propres à l'application industrielle à laquelle elle prend part et avoir certains comportements. Les scores doivent rester bornés entre 0 et 100 afin de s'accorder avec le fonctionnement de l'agent conversationnel que nous cherchons à améliorer. Dans le cas où il y a le même nombre de retours utilisateurs positifs et négatifs sur une réponse, le score de la réponse ne doit pas être modifié, car nous considérons alors qu'il n'y a pas d'accord entre les évaluateurs et que de ce fait leurs retours sont difficilement exploitables. Un vote utilisateur doit avoir un impact important sur le score de la réponse afin de pouvoir faire remonter le plus vite possible une bonne réponse en première position. Enfin, la fonction doit prendre en compte l'accord entre les utilisateurs sur la pertinence de cette réponse. L'impact sur le score ne doit pas être le même pour une réponse où les utilisateurs sont majoritairement d'accord, et une réponse provoquant un désaccord (un grand nombre de retours utilisateurs en opposition).

Le $NouveauScore$ est défini par :

si $DiffRetours \leq 0$, $NouveauScore = \frac{AncienScore}{e^{-Accord*DiffRetours}}$

sinon, $NouveauScore = 100 - 2 * (100 - AncienScore) + \frac{2*(100-AncienScore)}{1+e^{-Accord*DiffRetours}}$

avec r_+ = nombres de retours positifs, r_- = nombres de retours négatifs,

$$Accord = \frac{|r_+ - r_-|}{r_+ + r_-}$$

$$DiffRetours = r_+ - r_-$$

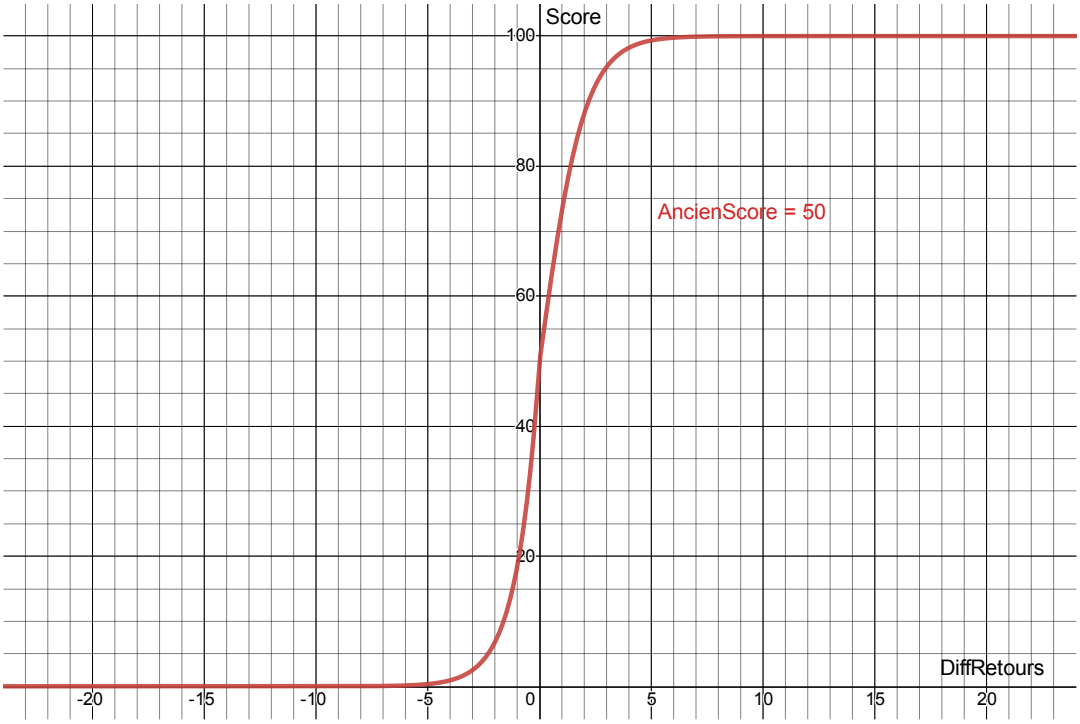

FIGURE 2 – Exemple de la fonction de modification du score

Comme l'illustre la Figure 2, les retours de cette fonction sont bien bornés entre 0 et 100. La nature exponentielle de la fonction induit bien que les premiers votes utilisateurs impactent fortement les scores. Une réponse jugée pertinente par les utilisateurs verra son score réévalué à la hausse et une réponse jugée non pertinente verra son score réévalué à la baisse. Si les réponses apportées par le système sont jugées peu pertinentes pour une question donnée, en ayant dégradé le score associé à ces réponses, le système présentera alors les réponses qui suivent dans la liste ordonnée qu'il produit. La liste des réponses potentielles sera ainsi progressivement parcourue. La fonction contient bien une identité, ne modifiant pas le score d'une réponse s'il n'y a pas de différence entre le nombre de retours positifs et négatifs qu'elle a reçue (ou si elle n'a reçue aucun retour utilisateur).

Le coefficient nommé « accord » impacte la pente de cette fonction : plus l'écart est faible entre un grand nombre de retours positifs et négatifs, moins la fonction d'activation est pentue et donc moins ces retours impactent le score de la réponse. Le parti pris étant de ne pas encourir le risque de modifier le score d'une réponse suscitant un important désaccord entre les utilisateurs.

Notons que la fonction a été divisée en deux parties pour des raisons de simplification de gestion au niveau de l'implémentation de l'agent conversationnel sur lequel s'est greffé l'approche proposée dans cet article.

3 Évaluation

Étant donné que l'approche proposée dans cet article s'inscrit dans une démarche scientifique et est aussi déployée dans des conditions d'utilisation réelles, nous avons dû évaluer certains aspects de la méthode. Dans un premier temps, nous avons dû vérifier le fait que l'expérience de l'utilisateur n'était pas dégradée. Ensuite, nous avons dû entraîner et sélectionner le classifieur de questions équivalentes le plus performant possible. Enfin, nous avons évalué les gains de cette approche.

3.1 Performance temporelle

L'approche proposée rajoutant un processus entre le moment où l'utilisateur pose sa question et celui où il obtient sa réponse, la durée d'exécution du processus est un paramètre critique. Nous avons tout d'abord identifié quelles étapes pouvaient être chronophages. La fonction changeant le score étant relativement simple d'un point de vue mathématique, ce n'est pas une étape prenant du temps. Le module retournant les questions similaires est déployé sous forme de service pouvant être appelé par l'agent conversationnel. Après évaluation, nous considérerons que le temps de réponse de ce service est égal à la somme du temps d'extraction des caractéristiques des questions et du temps d'exécution du classifieur.

Après avoir optimisé la durée d'exécution de cette étape nous avons réalisé une évaluation du temps de réponse de l'agent conversationnel. Pour cela nous avons posé au système les 10 questions les plus fréquemment utilisées par les utilisateurs et avons réalisé la moyenne des temps de réponses du système. Le temps de réponse moyen sur ces 10 réponses est d'environ 936 ms. Le temps de réponse maximal durant cette évaluation est de 1026ms. Le critère du temps de réponse maximal de l'agent conversationnel étant fixé à 2s, nous pouvons considérer que notre approche ne nuit pas à l'expérience de l'utilisateur en terme de temps d'attente de réponse.

3.2 Modèle d'équivalence entre questions

Dans le but d'évaluer le meilleur classifieur, nous avons évalué les performances des modèles suite à leurs entrainements. Nous avons pu évaluer la précision, le rappel et le f1-score de nos classifieurs sur le sous-ensemble de données annotées par les experts que nous avions gardé pour l'évaluation. Ces métriques sont évaluées pour chacune des classes prédites par nos modèles : questions équivalentes et non-équivalentes. Nous rappelons que le critère de sélection est le f1-score, rapport entre la précision et le rappel, sur la classe des questions équivalentes.

Nom du modèle	Classe	Précision	Rappel	F1-Score
Gradient stochastique	non-équivalentes	0.83	0.96	0.89
Gradient stochastique	équivalentes	0.89	0.60	0.72
Régression logistique	non-équivalentes	0.87	0.97	0.92
Régression logistique	équivalentes	**0.93**	0.7	0.80
Machines à vecteurs de support	non-équivalentes	0.93	0.94	0.93
Machines à vecteurs de support	équivalentes	0.87	0.85	0.86
Extreme Gradient Boosting	non-équivalentes	0.97	0.94	0.96
Extreme Gradient Boosting	équivalentes	0.88	**0.95**	**0.92**

TABLE 3 – Evaluation des classifieurs

Le classifieur proposant les meilleurs performances est un classifieur basé sur l'algorithme XGBoost (Chen & Guestrin, 2016) utilisant une grande majorité des caractéristiques de questions que nous avions explorées. Les performances de ce classifieur sont particulièrement élevées dans la détection des petites variations entre questions comme les fautes d'orthographe ou les fautes de frappe. Notons que le classifieur propose aussi le meilleur f1-score sur la classe des questions non-équivalentes même si cela n'impacte pas directement l'approche que nous proposons.

Remarquons que le classifieur proposant la meilleure précision sur la classe des questions équivalentes est celui basé sur une régression logistique. Nous ne l'avons pas retenu car son taux de rappel était trop faible.

Nous avons aussi exploré des modèles neuronaux qui retournaient de bons résultats, mais le gain en performance ne justifiait pas la perte au niveau du temps d'exécution.

3.3 Gain de l'approche

Le travail rapporté dans cet article est le fruit d'une réflexion ayant eu pour objectif d'améliorer les performances d'agents conversationnels déjà déployés auprès d'utilisateurs. Antérieurement à ce travail, les retours utilisateurs étaient pris en compte de façon rudimentaire, n'impactant qu'à la marge la qualité des réponses apportées par le système conversationnel. En effet, les retours utilisateurs n'étaient pris en compte que pour les questions posées précédemment de façon identique. Ainsi, une erreur de typographie empêche toute récupération des retours utilisateurs précédents. De plus, un retour utilisateur ne faisait que incrémenter (retour positif) ou décrémenter (retour négatif) de 1 le score de la réponse.

Afin d'évaluer le gain de notre approche nous avons récupéré un corpus créé par des experts du domaine de connaissance sur lequel évolue le système. Ce corpus comprend plus de 250 questions techniques propres au domaine de l'entreprenariat et la liste des réponses que retourne le système. Pour chacune des questions la position de la réponse attendue dans cette liste est de plus renseignée quand c'est possible par les experts.

En partant de l'hypothèse que les retours d'utilisateurs sont toujours corrects, nous avons simulé l'évolution de la précision moyenne (ici le pourcentage de questions issues du corpus où l'agent propose la bonne réponse directement à l'utilisateur) de l'agent conversationnel au fil des retours utilisateurs.

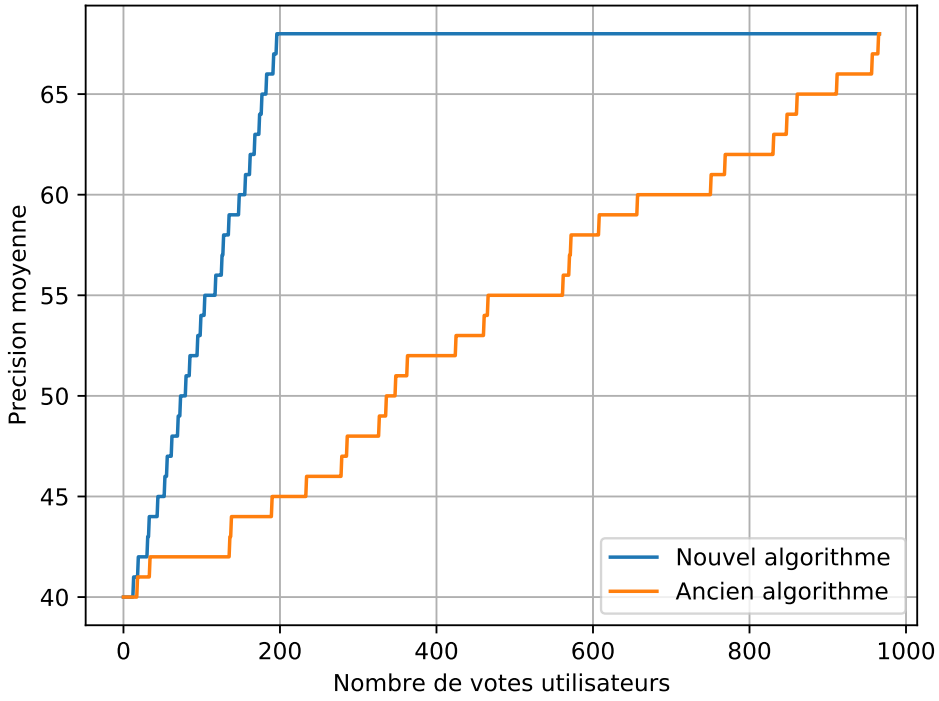

FIGURE 3 – Comparaison de la précision moyenne de l'agent conversationnel avec ou sans l'approche proposée

Comme nous pouvons le voir sur les résultats ci-dessous, notre approche permet de faire remonter en première position les bonnes réponses, améliorant la précision de l'agent, et cela beaucoup plus rapidement que dans l'approche précédemment utilisée.

4 Conclusion

Dans cet article, nous avons proposé une approche permettant d'améliorer les performances d'un agent conversationnel après son déploiement. En effet, lorsqu'un utilisateur pose une question au système, nous prenons en compte les retours que les utilisateurs ont fait sur des réponses à des questions équivalentes. Nous proposons donc la réponse la plus pertinente en utilisant l'expérience accumulée par l'agent.

Cette prise en compte des retours utilisateurs se fait « à la volée » entre le moment où l'utilisateur pose sa question et le moment où le système lui répond. L'approche peut se greffer par dessus tout type d'algorithme de rapprochement sémantique et fonctionne donc même quand cet algorithme ne prévoit pas de ré-entraînement. De plus, dans le système proposé les retours utilisateurs l'impactent immédiatement. Tandis que dans les approches où les systèmes se ré-entraînent, la prise en compte d'un retour utilisateur ne se fait qu'au moment du prochain ré-entraînement.

L'approche proposée utilise un modèle de détection d'équivalence entre questions. Quand un uti-

lisateur pose une question, l'agent peut récupérer des retours sur des réponses proposées pour des variantes de cette question. Cela permet au système de profiter encore plus de son expérience.

L'amélioration du système est liée à la qualité des retours des utilisateurs. En cas de désaccord entre les utilisateurs, nous avons choisi d'atténuer leur impact sur la réponse du système. Nous pouvons imaginer une approche complémentaire, dans laquelle un superviseur expert viendrait départager le désaccord. Dans d'autres approches, un désaccord entre utilisateurs pourrait brouiller l'apprentissage en accumulant des exemples contradictoires.

Finalement, nous pensons qu'il serait intéressant d'identifier au moment de la récupération des questions équivalentes, les couples questions-réponses ayant un nombre conséquent de retour, mais dans des directions opposées. Par exemple, un couple ayant 5 retours positifs et un autre ayant 7 retours négatifs. Nous pouvons considérer que l'accord fort entre les utilisateurs sur ces couples signifie que la réponse évaluée est la bonne pour l'une des questions mais une mauvaise pour l'autre. Or ces deux questions ont été classifiées comme étant équivalentes, une variante de la même question. Nous pourrions utiliser ces contre-exemples afin d'améliorer notre classifieur.

Références

AGIRRE E., CER D., DIAB M. & GONZALEZ-AGIRRE A. (2012). SemEval-2012 task 6 : A pilot on semantic textual similarity. In **SEM 2012 : The First Joint Conference on Lexical and Computational Semantics – Volume 1 : Proceedings of the main conference and the shared task, and Volume 2 : Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, p. 385–393, Montréal, Canada : Association for Computational Linguistics.

ALIANNEJADI M., ZAMANI H., CRESTANI F. & CROFT W. B. (2019). Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, p. 475–484, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3331184.3331265.

BELKIN N. J., COOL C., STEIN A. & THIEL U. (1995). Cases, scripts, and information-seeking strategies : On the design of interactive information retrieval systems. *Expert Systems with Applications*, **9**(3), 379–395. DOI : https ://doi.org/10.1016/0957-4174(95)00011-W.

BOTTOU L. (2010). Large-scale machine learning with stochastic gradient descent. *Proc. of COMPSTAT*. DOI : 10.1007/978-3-7908-2604-3_16.

CAMPOS J. A., CHO K., OTEGI A., SOROA A., AGIRRE E. & AZKUNE G. (2020a). Improving conversational question answering systems after deployment using feedback-weighted learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, p. 2561–2571, Barcelona, Spain (Online) : International Committee on Computational Linguistics. DOI : 10.18653/v1/2020.coling-main.230.

CAMPOS J. A., OTEGI A., SOROA A., DERIU J., CIELIEBAK M. & AGIRRE E. (2020b). DoQA - accessing domain-specific FAQs via conversational QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7302–7314, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.652.

CHEN T. & GUESTRIN C. (2016). Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, p. 785–794, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/2939672.2939785.

CHRISTMANN P., SAHA ROY R., ABUJABAL A., SINGH J. & WEIKUM G. (2019). Look before you hop : Conversational question answering over knowledge graphs using judicious context expansion. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, p. 729–738, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3357384.3358016.

DIAS G., Éd. (2015). *Actes de TALN 2015 (Traitement automatique des langues naturelles)*, Caen. ATALA, HULTECH.

GAO J., GALLEY M. & LI L. (2019). Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, **13**(2-3), 127–298. DOI : 10.1561/1500000074.

HANCOCK B., BORDES A., MAZARE P.-E. & WESTON J. (2019). Learning from dialogue after deployment : Feed yourself, chatbot ! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 3667–3684, Florence, Italy : Association for Computational Linguistics. DOI : 10.18653/v1/P19-1358.

HOI S. C. H., SAHOO D., LU J. & ZHAO P. (2018). Online learning : A comprehensive survey.

KUSNER M. J., SUN Y., KOLKIN N. I. & WEINBERGER K. Q. (2015). From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 957–966 : JMLR.org.

LI J., MILLER A. H., CHOPRA S., RANZATO M. & WESTON J. (2016). Dialogue learning with human-in-the-loop. *CoRR*, **abs/1611.09823**.

LIU B., TÜR G., HAKKANI-TÜR D., SHAH P. & HECK L. (2018). Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long Papers)*, p. 2060–2069, New Orleans, Louisiana : Association for Computational Linguistics. DOI : 10.18653/v1/N18-1187.

PRABOWO D. A. & BUDI HERWANTO G. (2019). Duplicate question detection in question answer website using convolutional neural network. In *2019 5th International Conference on Science and Technology (ICST)*, volume 1, p. 1–6. DOI : 10.1109/ICST47872.2019.9166343.

QU C., YANG L., CHEN C., QIU M., CROFT W. B. & IYYER M. (2020). Open-retrieval conversational question answering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, p. 539–548, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3397271.3401110.

QU C., YANG L., CROFT W. B., ZHANG Y., TRIPPAS J. R. & QIU M. (2019). User intent prediction in information-seeking conversations. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, CHIIR '19, p. 25–33, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/3295750.3298924.

RAJPURKAR P., ZHANG J., LOPYREV K. & LIANG P. (2016). SQuAD : 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 2383–2392, Austin, Texas : Association for Computational Linguistics. DOI : 10.18653/v1/D16-1264.

REDDY S., CHEN D. & MANNING C. D. (2019). CoQA : A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, **7**, 249–266. DOI : 10.1162/tacl_a_00266.

TELLEX S., KATZ B., LIN J., FERNANDES A. & MARTON G. (2003). Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International*

ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03, p. 41–47, New York, NY, USA : Association for Computing Machinery. DOI : 10.1145/860435.860445.