L'apprentissage d'ordonnancement pour l'appariement de questions

Camille Pradel, Baptiste Chardon,
Dominique Laurent, Sophie Muller, Patrick Séguéla
Synapse Développement, 5 rue du Moulin-Bayard, 31000 Toulouse,
{ camille.pradel, baptiste.chardon, dlaurent, sophie.muller,
patrick.seguela }@synapse-fr.com

Résumé : Cet article présente une approche permettant à un utilisateur d'interroger une base de connaissances type FAQ, c'est-à-dire un ensemble de questions et leurs réponses respectives rédigées en langue naturelle. Le composant présenté dans cet article apparie la question de l'utilisateur à une ou plusieurs questions de la base de connaissances. Pour cela, nous utilisons un composant déjà existant d'analyse de questions, capable de sélectionner un ensemble de questions candidates proches de la question utilisateur, et de produire des traits propres à chaque couple (question utilisateur, question candidate). Ce composant est chaîné à un modèle permettant l'ordonnancement des questions candidates, qui est appris automatiquement de façon supervisée, une partie seulement du corpus d'apprentissage étant annotée manuellement, et le reste grâce à des règles ad-hoc. Ces travaux reprennent les résultats d'un domaine de recherche récent, l'apprentissage d'ordonnancement (*Learning to Rank*), et les adaptent à une application industrielle innovante, l'appariement de questions comme paradigme d'accès à la connaissance. Une expérimentation évalue sur des données issues d'un système en production la qualité de chacune des phases d'apprentissage.

Mots-clés : Interface en langue naturelle, FAQ, *Learning to rank*, apprentissage actif, recherche d'information.

1 Introduction

Une des tâches les plus populaires en Recherche d'Information (RI) consiste à retrouver un ensemble de documents pertinents vis-à-vis d'un besoin en information exprimé dans une requête. Pour cela, des approches efficaces existent depuis de nombreuses années ; elles sont fondées sur des heuristiques et n'exploitent généralement pas des modèles appris automatiquement à partir de corpus d'entrainement. Bien que simples et élégantes sur le papier, ces approches nécessitent pour être déployées en production une phase de réglage lourde et fastidieuse, tout particulièrement lorsque le nombre de paramètres à prendre en compte augmente (les moteurs de recherche du web prennent en compte plusieurs dizaines de paramètres et doivent sans cesse en intégrer de nouveaux pour rester compétitifs). L'idée d'apprendre automatiquement les modèles de classement prenant en compte tous ces paramètres devient de plus en plus séduisante au fur et à mesure que la mise en œuvre des algorithmes traditionnels se complexifie, d'autant plus que l'acquisition des données d'entrainement est, dans ce cas, simple et peu onéreuse (le choix de l'utilisateur parmi les résultats qui lui sont retournés étant un très bon indice de la pertinence de ces résultats). Yahoo ! et Microsoft ont successivement libéré des jeux de données et organisé des compétitions pour stimuler les recherches dans ce domaine.

Dans cet article, nous voulons exploiter ces mêmes méthodes et les appliquer à un domaine différent, celui de l'appariement de questions. L'objectif du système présenté dans la suite est d'apparier une question exprimée par un utilisateur à une ou plusieurs questions cibles issues d'une base de connaissances. Dans ce contexte, nous appelons base de connaissances un ensemble de questions cibles et leurs réponses respectives rédigées en

langue naturelle, une même réponse pouvant éventuellement être associée à plusieurs formulations de la même question, permettant ainsi de surmonter plus facilement la variabilité de la langue. Cet appariement peut notamment être utilisé pour suggérer les questions cibles dans un champ de recherche avec autocomplétion, ou bien au sein d'un agent conversationnel. Plus généralement, l'appariement de questions constitue une alternative aux questions-réponses particulièrement pertinente pour l'accès au savoir en milieu industriel, dont les avantages sont discutés en section 2.

Le système développé exploite le composant d'analyse de requête du moteur de question-réponse de Synapse développement. Ce composant est capable de sélectionner un ensemble de questions de la base de connaissances, appelées questions candidates dans la suite, qui sont sémantiquement proches de la question utilisateur. Il extrait aussi certaines caractéristiques de ces questions, comme leur type (recherche d'une date de naissance ou de mort, d'une durée, d'une ville, d'une personnalité…), leur objet (ce sur quoi elles portent) et les mots-clés qui les composent. Ce composant produit également un score pour chacune des questions appariées. De ce score, on pourrait immédiatement déduire un ordonnancement. Cependant, le système d'origine ayant pour objectif de chercher dans un texte la réponse à une question, ce score, bien que significatif, ne prend pas en compte certains des aspects essentiels à l'appariement de questions (par exemple, est-ce que les deux questions comparées sont de même type ?).

C'est pourquoi nous avons défini dans un premier temps un ensemble de règles permettant de (ré)ordonnancer les résultats produits par ce composant. Le système ainsi obtenu fonctionne suffisamment bien pour être utilisé en production. Mais, dans le but d'éviter d'avoir à maintenir un système de règles toujours plus complexe, nous voulons apprendre un modèle qui produit dans un premier temps des sorties similaires à ce système de règles puis qui saura se raffiner et s'adapter aux usages en exploitant les retours utilisateurs pour évoluer.

Nous justifions la pertinence de notre approche dans la section 2 et donnons un aperçu des méthodes existantes en apprentissage d'ordonnancement en section 3. Puis nous décrivons les travaux réalisés en section 4 et présentons en section 5 les résultats des évaluations. Enfin, nous concluons et présentons quelques perspectives en section 6.

2 Vers un nouveau paradigme d'accès à la connaissance

Dans cette section, nous justifions notre choix d'exploiter l'appariement de questions en langue naturelle plutôt que l'approche de questions-réponses traditionnelle par trois éléments : la difficulté moindre de la tâche d'appariement, le besoin de réassurance de l'utilisateur et le format des connaissances actuellement disponibles dans le milieu industriel.

2.1 Limites des interfaces entre langue naturelle et texte

Les interfaces entre la langue naturelle et les textes, ou systèmes de questions-réponses, ont fait l'objet de recherches très actives dans les années 2000. Ces approches exploitent une collection de documents comme source de connaissances et cherchent directement la ou les réponses à une question utilisateur dans ces documents. Le composant d'analyse de questions utilisé dans les expérimentations dans la suite de l'article est issu de *QRISTAL*, le système de question-réponse de *Synapse Développement* (Laurent & al., 2005).

D'après (Hirschman & Gaizauskas, 2001), le processus d'interprétation de la grande majorité de ces systèmes se divise en deux étapes :

1. identifier le type de l'objet de la requête ; une bonne reconnaissance du type de l'objet est primordiale, et de nombreux travaux se sont penchés sur le problème, en définissant des hiérarchies de types à partir du type de la réponse attendue (Moldovan & al., 1999; Hovy & al., 2000; Wu & al., 2003; Srihari & Li, 1999) ;
2. déterminer les contraintes additionnelles exprimées ; ces contraintes sont le plus souvent issues des relations syntaxiques et sémantiques entre les termes de la

requête en langue naturelle (Moldovan & al., 2002; Litkowski, 2000; Attardi & al., 2002).

En comparaison aux interfaces entre langue naturelle et bases de données qui les ont précédés, les systèmes de question-réponse sont complexes, du fait de la non-restriction du domaine et de la forme beaucoup moins contrainte et plus variable de la source de connaissances. Ils intègrent de nombreuses sous-tâches, parmi lesquelles on peut citer la reconnaissance d'entités nommées, l'extraction de relations, la résolution de coréférences et la désambiguïsation.

Un sous-ensemble de ces systèmes, apparu un peu plus tard, considère le web dans son intégralité comme corpus de documents. La tâche reste la même mais certains problèmes, comme le passage à l'échelle et l'hétérogénéité des données, prennent toute leur importance. Pour faire face à l'immensité du web, la majorité des systèmes proposés dans la littérature exploite des moteurs de recherche « classiques » par mots-clés, comme Google. La requête utilisateur est d'abord transformée en une ou plusieurs requêtes compatibles avec le moteur de recherche choisi, puis ces requêtes sont exécutées sur le web, et enfin les réponses sont extraites des documents les plus pertinents renvoyés par le moteur de recherche.

Les recherches visant à la résolution de cette tâche ont été largement orientées, de 1999 à 2007, par la compétition question-réponse sans restriction de domaine (*open domain question answering*) de la conférence sur l'extraction de données dans les textes (*TREC - Text REtrieval Conference*). Les systèmes développés durant ces années ont obtenu d'excellents résultats. Les recherches dans le domaine sont beaucoup moins actives depuis quelques années, et on tend à considérer ce problème comme résolu.

On constate cependant que la mise en place réelle de tels systèmes auprès d'utilisateurs finals n'a jamais été réalisée, si ce n'est pour des besoins de démonstration. Leur développement, notamment à l'échelle du web, se heurte à des verrous de taille (passage à l'échelle, gestion du bruit, identification des doublons...), et les solutions trouvées nécessitent d'importants efforts d'implémentation. Il nous semble que les systèmes proposés au cours des compétitions TREC souffrent de suradaptation aux données des compétitions et que leur évolution dans un contexte industriel n'est pas triviale.

2.2 Utilisabilité des interfaces en langue naturelle

La pertinence des interfaces en langue naturelle a été mise en cause plusieurs fois dans la littérature, sans jamais pour autant mener à une conclusion claire (Chakrabarti, 2004; Dekleva, 1994; Desert, 1993; Thompson & al., 2005).

Plusieurs études (Dittenbach & al., 2003; Reichert & al., 2005) ont voulu analyser l'intérêt des utilisateurs finals pour les requêtes en langue naturelle, par rapport aux requêtes par mots-clés. Les auteurs de (Dittenbach & al., 2003) ont ainsi développé une interface test de requêtes libres dans le domaine du tourisme, et récolté 1425 requêtes spontanées d'utilisateurs. 57,05% des requêtes étaient des phrases complètes exprimées en langue naturelle et grammaticalement correctes ; 21,26% étaient des fragments de questions, du type « double room for two nights in Vienna » ; 21,69% étaient des requêtes mots-clés. D'après les auteurs, les interfaces en langue naturelle se montrent particulièrement utiles lorsque le public visé est hétérogène, comme dans le cas de la plate-forme utilisée. Les auteurs de (Reichert & al., 2005) ont demandé à des étudiants d'utiliser deux versions différentes d'un outil de question-réponse pour le *e-learning*, *CHESt* (Linckels & Meinel, 2005). Dans une version, les requêtes s'expriment sous forme de mots-clés ; dans l'autre, elles s'expriment en langue naturelle. 76% des étudiants ont déclaré préférer la version mots-clés mais concèdent également qu'ils utiliseraient plus volontiers la version langue naturelle si ils savaient pouvoir obtenir de meilleurs résultats.

Dans (Kaufmann & Bernstein, 2010), les auteurs évaluent l'utilisabilité des interfaces permettant aux utilisateurs d'accéder à des connaissances par le biais de la langue naturelle. Ils identifient trois principaux problèmes liés à ce type d'interface :

- l'*ambiguïté* de la langue naturelle est le plus évident, l'*ambiguïté linguistique* ─ une même expression en langue naturelle peut être interprétée de différentes façons ─ va de pair avec la *variabilité linguistique* ─ une même idée peut s'exprimer de différentes façons en langue naturelle.
- la *barrière adaptative* (*adaptivity barrier*) rend les interfaces présentant de bonnes performances de recherche peu portables ; ces systèmes sont la plupart du temps spécifiques au domaine et, par conséquent, difficiles à adapter à de nouveaux contextes. L'ensemble des connaissances ontologiques contenues dans une base de connaissances est vu comme un moyen réaliste de pallier ce problème.
- le *problème d'habitabilité* (*habitability problem*) selon lequel l'utilisateur final peut se sentir perdu face à une trop grande liberté dans l'expression de sa requête. La plupart des interfaces en langue naturelle n'expliquent pas comment exprimer ses requêtes ou quel type d'information peut être demandé ; en conséquence, l'utilisateur peut exprimer des requêtes au-delà des capacités du système (que ce soit du point de vue du besoin en information ou de l'expression de la requête) ou, pire encore, le système peut mal interpréter la requête et l'utilisateur peut ne pas s'en rendre compte et considérer une réponse inappropriée comme satisfaisante.

La contribution principale de (Kaufmann & Bernstein, 2010) est inspirée de ce dernier problème : l'*hypothèse d'habitabilité* prétend qu'une interface en langue naturelle, pour présenter la meilleure utilisabilité, devrait imposer une certaine structure à l'utilisateur dans le but de l'orienter lors du processus de formulation de la requête. Ainsi le « syndrome de la feuille blanche » devrait être évité. Cependant, la contrainte structurelle ne doit pas être trop restrictive, le risque étant d'aliéner l'utilisateur.

Pour soutenir cette hypothèse, les auteurs décrivent l'étude d'utilisabilité qu'ils ont conduite auprès d'utilisateurs. Quatre interfaces de requêtes ont été développées et confrontées à 48 utilisateurs. Les auteurs proposent de situer chaque interface sur un *continuum de formalisation* (*formality continuum*) en fonction de la nature de l'interaction avec l'utilisateur final. Les approches complètement LN sont à une extrémité de ce continuum, alors que les langages formels de requêtes sont à l'autre extrémité. Les résultats de cette étude montrent clairement que, conformément à l'hypothèse d'habitabilité, les utilisateurs se sentent le plus à l'aise en utilisant Querix et son langage de requêtes légèrement contraint.

Ce point de vue est renforcé par des travaux plus récents présentés dans (Damljanović et al., 2013) qui exploitent des retours utilisateurs et des fenêtres de clarification pour augmenter l'expérience de l'utilisateur.

Nous retenons de ces résultats le besoin d'un retour de l'interface vers l'utilisateur afin de rassurer ce dernier quant aux informations qui lui sont retournées en réponse à sa requête.

2.3 Bases de connaissances sous forme de FAQ

Il est possible de trouver chez des grands groupes industriels, notamment aéronautiques et pharmaceutiques, des bases de connaissances structurées, par exemple sous forme de graphe, avec des schémas cohérents et des données fiables respectant ces schémas. Ces ressources critiques ne sont généralement pas accessibles publiquement.

Les entreprises de taille plus modeste n'investissent en général pas dans la construction de telles ressources, très onéreuse, et la grande majorité des connaissances métiers des entreprises reste à ce jour encore exprimée sous forme textuelle.

Parmi ces ressources textuelles, une partie est rédigée sous forme de FAQ (pour *Frequently Asked Questions*), c'est-à-dire de couples (question, réponse) exprimés en langue naturelle. Ce format est en effet traditionnellement utilisé pour présenter des informations à l'utilisateur final d'un service. Il est censé permettre de trouver plus rapidement une réponse à un besoin précis en information ; le lecteur se contente de parcourir les questions jusqu'à trouver celle qui l'intéresse et évite ainsi une lecture exhaustive du document. Cette lecture reste cependant linéaire et, si la FAQ devient trop importante, une simple recherche peut devenir très laborieuse. Certaines entreprises, typiquement les éditeurs de logiciels, ont

construit des FAQ très importantes (souvent appelées bases de connaissances) ; elles peuvent être hiérarchisées par catégories de questions, mais, même ainsi, leur parcours manuel n'a plus de sens, car trop inefficace.

Il existe donc toujours un besoin de valoriser ces connaissances exprimées sous forme de FAQ et de faciliter leur accès par un utilisateur final, et c'est à ce besoin que l'approche que nous développons dans ce papier veut répondre.

2.4 L'appariement de questions

Pour les raisons exprimées dans les sous-sections précédentes, nous travaillons depuis peu sur l'évolution de notre moteur de question-réponse vers un nouveau mode d'interrogation des connaissances. Celui-ci est fondé sur l'appariement de questions d'une base de connaissances à la question de l'utilisateur.

Dans cette approche, l'expression du besoin en information de l'utilisateur peut par exemple se faire au travers d'un champ de recherche avec une autocomplétion suggérant des questions proches. L'utilisateur est ainsi confronté à un mode d'interaction qui lui est familier, il bénéficie d'un retour du système et porte ainsi une plus grande confiance aux informations qui lui sont retournées. S'il sélectionne une des questions suggérées, équivalente ou proche de sa question d'origine, il est certain que la réponse qui lui est affichée répond à la question sélectionnée. S'il renseigne sa propre question jusqu'au bout et la valide, c'est la réponse à la question la plus proche qui lui est soumise, mais l'interface devrait afficher la question appariée.

Selon l'hypothèse d'habitabilité de (Kaufmann & Bernstein, 2010), introduite ci-dessus, l'interface que nous proposons, située au milieu du continuum de formalité, offre un *feedback* qui permet d'augmenter la réassurance de l'utilisateur et de maximiser son expérience.

3 Learning to rank

L'apprentissage d'ordonnancement (*learning to rank*) est un domaine de recherche relativement récent ; l'objectif est de développer des algorithmes d'apprentissage pour des tâches d'ordonnancement.

Le principal domaine d'application est la RI (Liu, 2009), mais des travaux ont également été menés dans le traitement automatique des langues, pour l'ordonnancement de traductions potentielles d'une phrase en traduction automatique (Li, 2014). Une vue d'ensemble des principales approches existantes est donnée dans (Li, 2011).

Les travaux réalisés à ce jour sont en grande majorité issus de participations aux compétitions organisées par Yahoo ! et Microsoft, et ont été évalués sur les données fournies dans le cadre de ces compétitions. Les résultats actuels en apprentissage d'ordonnancement sont donc liées aux problématiques du domaine de la Recherche d'Informations (RI). Ces méthodes d'apprentissage ont déjà été expérimentées pour la recherche au sein de bases de connaissances type FAQ (Surdeanu & al., 2008 ; Bunescu & Huang, 2010)

3.1 Les catégories d'approches

L'ensemble des méthodes proposées jusque-là se divise en trois catégories. Les approches points par points (*pointwise approaches*) transforment le problème d'ordonnancement en un problème d'apprentissage automatique classique, comme la classification ou la régression ; il est ainsi possible d'exploiter les algorithmes de la littérature liés à ces problèmes. Les principales approches dans ce domaine comptent Subset Ranking (Cossock & Zhang, 2006), *McRank* (Li & al., 2007), *Prank* (Crammer & Singer, 2001), et *OC SVM* (Shashua & Levin, 2002).

Dans les approches par paires (*pairwise approaches*), l'ordonnancement est associé à un ensemble de classification par paires ou de régression par paires ; là encore, les algorithmes d'apprentissage existants peuvent être exploités. *Ranking SVM* (Herbrich & al., 1999), *RankBoost* (Freund & al., 2003), *RankNet* (Burges & al., 2005), *GBRank* (Zheng et al., 2008), *IR SVM* (Cao et al., 2006), et *LambdaMART* (Wu et al., 2010) font partie de ces approches.

La dernière catégorie de méthodes est spécifique à la tâche d'ordonnancement. Ces approches sont les seules à exploiter la structure de groupe ordonné des données d'apprentissage. C'est dans ce cas l'ordonnancement entier qui est considéré comme une instance d'apprentissage. Les principales approches de cette catégorie sont *ListNet* (Cao et al., 2007), *ListMLE* (Xia et al., 2008), *AdaRank* (Xu & Li, 2007), *SVM MAP* (Yue et al., 2007), et *Soft Rank* (Taylor et al., 2008).

3.2 Métriques d'évaluation

Tout comme en RI, les métriques d'évaluation représentent un aspect essentiel de l'apprentissage d'ordonnancement. Elles définissent la fonction objectif à optimiser lors des phases d'apprentissage et de validation. L'objectif de ces mesures est d'exprimer à quel point l'ordonnancement proposé par le système évalué est proche de l'ordonnancement de référence. Les métriques utilisées pour l'apprentissage d'ordonnancement sont à ce jour celles de la RI : Discounted Cumulative Gain (DCG) (Järvelin & Kekäläinen, 2002), Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), Kendall's Tau, Reciprocal Rank (RR) (Voorhees & Tice, 1999), Expected Reciprocal Rank (ERR) (Chapelle & al., 2009).

4 Construction du modèle d'ordonnancement

Comme expliqué plus haut, le modèle d'ordonnancement est construit en deux étapes :

1. Des règles simples prédéfinies sont utilisées pour produire des ordonnancements sur des questions appariées à un ensemble des questions utilisateurs issues d'un log de système en production permettant d'interroger une FAQ. On obtient ainsi un ensemble de couples (question utilisateur, ordonnancement de questions candidates), suffisamment important pour initier le modèle, qui intègre alors l'« intelligence » des règles définies précédemment.
2. En interagissant avec le système, les utilisateurs produisent des informations qui peuvent être exploitées pour raffiner le modèle. On considère en effet que lorsque l'utilisateur sélectionne une question dans la liste des suggestions qui lui sont proposées, cette question est la plus proche de son besoin en information initial, et on peut intégrer cette connaissance au modèle afin de favoriser la question sélectionnée lorsqu'une nouvelle question entrante présente les mêmes caractéristiques.

La figure 1 présente l'architecture générale du composant d'appariement de questions. Les deux cadres en pointillé représentent chacun une des phases d'apprentissage du modèle. Dans la phase *batch learning*, des ordonnancements issus de règles sont utilisés comme exemples. Puis dans la phase *active learning*, des informations sont recueillies auprès de l'utilisateur dans le but d'améliorer le modèle.

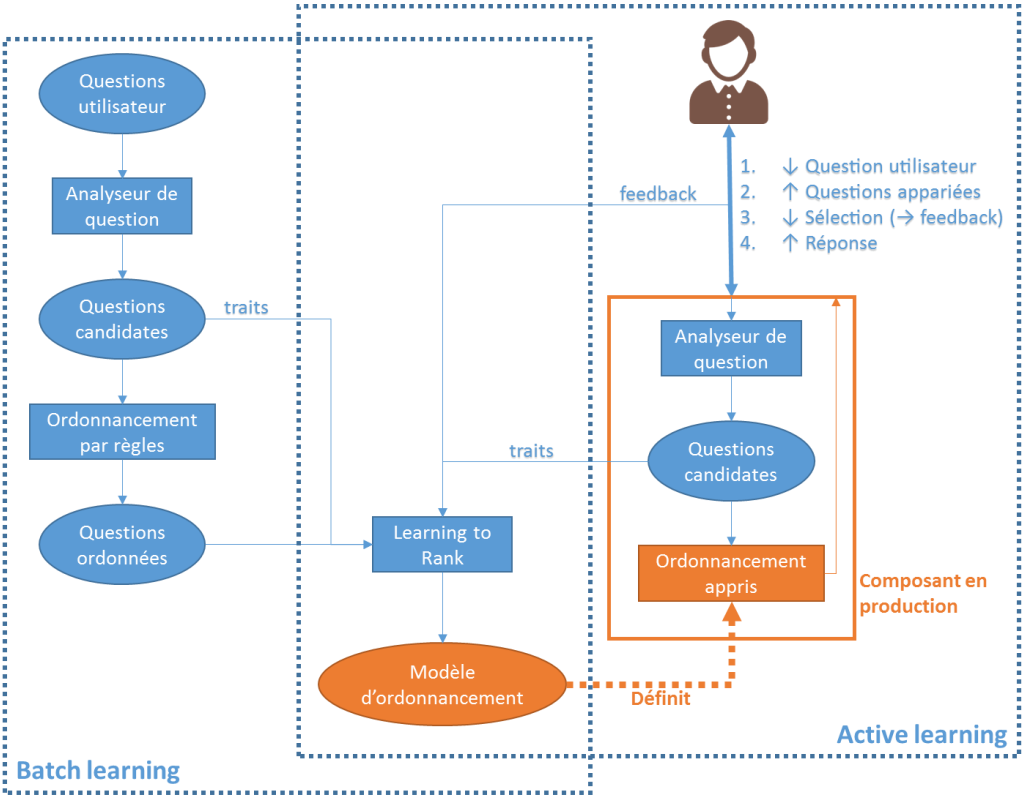

FIGURE 1 – Vue d'ensemble du système d'appariement de questions.

4.1 Génération du corpus d'apprentissage

Les logs que nous avons utilisés comportent 4855 requêtes exprimées par des utilisateurs. Il est important de préciser que ces 4855 ne sont pas toutes des questions. Dans certaines, le besoin en information est exprimé sous forme de mot-clés, comme par exemple « problème installation » ; d'autres n'expriment aucun besoin en information, on trouve par exemple des « bonjour », « merci », « au revoir », mais aussi des insultes ou d'autres entrées improbables, certainement dues à une mauvaise manipulation, par exemple « i ». Ces caractéristiques apportent une difficulté supplémentaire qui est abordée dans la section Perspectives.

Pour construire le corpus d'apprentissage, nous avons analysé chacune des 4855 requêtes afin d'obtenir, pour chaque requête, l'ensemble des questions candidates de la base de connaissances et les traits associés à chaque couple (question utilisateur, question candidate). Puis, sur ces 4855 ensembles de couples, 500 ont été ordonnés à la main pour simuler un retour utilisateur et les 4355 autres à l'aide du système de règles. Dans les expérimentations décrites plus bas, les trois quarts des données annotées à la main ont été utilisés pour l'apprentissage, le dernier quart a été conservé pour les tests.

Dans la littérature, pour exprimer le classement des requêtes candidates pour une même requête utilisateur, à chaque couple (question utilisateur, question candidate) est associée une note de pertinence, qui exprime à quel point le besoin exprimé dans la question candidate est proche de celui exprimé dans la question utilisateur. Cette note peut prendre une valeur entière de 0 à 4, 0 signifiant que les deux questions n'ont rien en commun, et 4 qu'elles sont extrêmement proches, voire identiques.

Nous avons utilisé cette même approche dans le but de simplifier la réexploitation des algorithmes existants. Lors de la génération des ordonnancements à partir des règles, la première question se voit attribuer la valeur 4, la deuxième 3, la troisième 2, la quatrième 1 et toutes les suivantes 0. Dans le cas où le composant d'analyse de requêtes ne retourne qu'une

seule requête candidate, nous lui attribuons la note 3 (cette décision a été prise après une observation qualitative d'un extrait des sorties de l'analyseur de question).

La constitution de l'ordonnancement manuel a été effectuée par un seul annotateur avec pour seule consigne de calquer la note utilisée dans les jeux de données publiés par Microsoft : « *The relevance judgments […] take 5 values from 0 (irrelevant) to 4 (perfectly relevant)* ».

4.2 Choix des traits

Pour constituer les données d'apprentissage, nous avons intégré l'ensemble des traits retournés par le composant d'analyse de question. Cela assure de donner à l'algorithme d'apprentissage tous les éléments pour intégrer l'intelligence codée dans les règles. Ces traits sont les suivants : 1. le score introduit plus haut qui représente la similarité entre la question utilisateur et la question candidate, 2. le nombre de mots-clés que la question candidate a en commun avec la question utilisateur, 3. le nombre de mots-clés distincts communs à la question candidate et à la question utilisateur, 4. un booléen indiquant si les deux questions sont de même type.

Nous avons également ajouté d'autres traits dans le but de permettre à l'algorithme d'apprendre des subtilités qui ne sont pas représentées dans les règles mais pourraient être déduites des feedbacks utilisateur :

- Quatre traits liés à la ressemblance entre les chaines de caractères de chaque question : 1. la distance de Levenstein entre les deux questions, 2. un booléen indiquant si les chaines de caractères sont identiques après neutralisation des accents et des majuscules, 3. un booléen indiquant si la chaine de la question utilisateur contient la chaine de la question candidate, 4. un booléen indiquant si la chaine de la question candidate contient la chaine de la question utilisateur,
- Deux traits visant à prendre en compte la « popularité » des questions de la base de connaissances, c'est-à-dire dans quelle mesure ces questions représentent les besoins en information généralement exprimés ; ces notes sont calculées sur l'ensemble des données d'entrainement au cours d'un pré-traitement : 1. un entier exprimant combien de fois une question apparaît dans une liste de questions candidates, 2. un entier exprimant combien de fois une question obtient la note de pertinence maximale.

4.3 Choix de la métrique

Pour choisir la métrique utilisée pour l'apprentissage (et pour l'évaluation présentée plus bas), nous considérons une intégration du système dans un champ de recherche avec une autocomplétion. Dans ce scénario, l'utilisateur veut se voir suggérer au plus vite une question qui correspond à son besoin en information. Il veut la réponse à sa question et rien de plus ; il n'est pas à la recherche de l'exhaustivité, comme lorsque l'on fait une recherche approfondie ou de la veille.

Nous avons pour objectif que l'utilisateur n'ait pas à consulter plus de trois suggestions avant de trouver celle correspondant à son besoin en information. Nous utilisons donc la mesure ERR@3 (Chapelle & al., 2009), qui représente la probabilité que l'utilisateur ait trouvé ce qu'il cherche dans les trois premières suggestions, tout en privilégiant les suggestions pertinentes tout en haut du classement.

4.4 Augmentation du corpus d'entrainement avec les questions de la base de connaissances

Le log de requêtes (et par conséquent le corpus d'entrainement) en notre possession étant de taille relativement réduite pour mettre en œuvre un apprentissage automatique, nous avons

voulu tester l'utilisation d'un artifice permettant de le gonfler légèrement avec des données fiables.

Nous avons pour cela augmenté notre corpus avec l'ensemble des questions de la base de connaissances, que l'on considère alors comme des questions utilisateurs. En sortie du système de règles, chacune de ces questions est logiquement associée à elle-même avec le score maximal, nous voulons ainsi permettre à l'algorithme d'apprentissage de prendre en compte les traits liés aux chaines de caractères décrits plus haut.

4.5 Augmentation de l'influence des données annotées manuellement

Les données annotées à la main sont présentes en moins grande quantité dans le corpus d'apprentissage et influencent donc beaucoup moins le modèle appris alors qu'elles sont censées être plus fiables que celles générées à partir des règles. Pour compenser cette sous-représentation, nous avons testé deux méthodes permettant d'augmenter leur impact sur le processus d'apprentissage.

La première méthode consiste simplement à suréchantillonner les données manuelles, en les reproduisant plusieurs fois à l'identique dans le corpus d'entrainement. Dans la seconde méthode, le corpus d'apprentissage reste le même (il contient donc les données issues des règles et celles annotées à la main en un seul exemplaire), mais nous contraignons l'algorithme d'apprentissage à considérer les données manuelles comme ensemble de validation. L'impact de ces deux méthodes est présentée dans la section suivante.

5 Évaluations

Pour les expérimentations, nous avons utilisé la bibliothèque RankLib[1], qui implémente plusieurs algorithmes d'apprentissage d'ordonnancement. Chaque algorithme cité dans la suite a été utilisé avec les paramètres définis par défaut dans la bibliothèque. D'une façon générale, nos efforts ne se sont pas concentrés sur l'optimisation des scores, notre objectif étant avant tout de montrer que le modèle s'améliore en prenant en compte les retours utilisateurs.

La table 1 montre les résultats obtenus pour l'ensemble des algorithmes testés et pour différentes configurations des données d'entrainement et de validation. De par la métrique d'évaluation choisie (ERR@3), les valeurs données dans cette section sont relativement basses en comparaison des métriques habituelles de la RI. En effet, comme expliqué plus haut, ce score représente la probabilité que l'utilisateur ait trouvé ce qu'il cherche dans les trois premières suggestions ; il n'est donc pas lié uniquement à la qualité de l'ordonnancement, mais aussi à la capacité du corpus à répondre au besoin en information. Pour mettre en évidence cette propriété de la métrique, nous avons calculé sa valeur maximale, c'est-à-dire la note qu'obtiendrait un oracle produisant un ordonnancement parfait.

Nous avons également construit une heuristique simple pour obtenir une *baseline*. Celle-ci renvoie toujours les trois mêmes suggestions, les plus populaires dans le corpus d'entrainement (la popularité est mesurée par le somme des notes attribuées). De par la très forte présence d'un nombre restreint de questions, cette *baseline* donne d'assez bons résultats, ce qui nous a incité à intégrer la notion de popularité des questions cibles dans les traits décrits plus haut.

Aucune tendance ne se dégage clairement de l'ajout des questions de la base de connaissances au corpus d'entrainement (deuxième colonne). Les résultats de l'apport des retours utilisateurs sont plus encourageants. Même si lorsque l'on ajoute simplement les données annotées à la main aux données d'entrainement, le score n'est amélioré que pour la moitié des algorithmes (troisième colonne), on constate une amélioration significative pour les deux algorithmes présentant les meilleures performances (Coordinate Ascent et

[1] https://sourceforge.net/p/lemur/wiki/RankLib/

LambdaMART, lignes en gras dans le tableau). De plus, la progression est plus marquée encore lorsque l'on donne artificiellement plus de poids à ces données. La quatrième colonne montre les résultats obtenus lorsque l'on fixe les données de validation comme étant les données annotées à la main. Enfin, les cinquième et sixième colonnes montrent les scores obtenus en dupliquant les données annotées à la main dans le corpus d'entrainement.

TABLE 1 – *Résultats obtenus pour l'ensemble des algorithmes testés et pour différentes configurations des données d'entrainement et de validation. Les valeurs montrent le score ERR@3 de chaque approche ; les scores en vert (resp. rouge) indiquent une amélioration (resp. dégradation)des résultats obtenus en augmentant le corpus d'apprentissage avec les retours utilisateurs.*

		apport des questions de la KB	apport des feedbacks utilisateurs			
				avec suréchantillonage		
données d'entrainement	règles	règles + questions kb	règles + questions kb + feedback		règles + questions kb + 3x feedbacks	règles + questions kb + 10x feedbacks
jeu de validation	20% du jeu d'entr.	20% du jeu d'entr.	20% du jeu d'entr.	feedback	20% du jeu d'entr.	20% du jeu d'entr.
MART	0,4319	0,4299	0,4299	0,4230	0,4293	0,422
RankNet	0,2623	0,2686	0,2794	0,2694	0,2626	0,2668
RankBoost	0,4147	0,4197	0,3908	0,3908	0,4179	0,3817
AdaRank	0,4495	0,4318	0,4203	0,4661	0,4407	0,4203
Coordinate Ascent	**0,4531**	**0,4547**	**0,4549**	**0,4549**	**0,4539**	**0,4547**
LambdaMART	**0,4548**	**0,4527**	**0,4575**	**0,4550**	**0,4845**	**0,4773**
ListNet	0,2603	0,2599	0,2597	0,2552	0,2597	0,2588
Random Forests	0,4216	0,4224	0,4218	0,4210	0,4227	0,4225
Popularité **(baseline)**	0,4030	0,3986	0,4023		0,4266	0,4514
Ordonnancement parfait **(upperline)**	0,6775					

6 Conclusion et perspectives

Nos expérimentations ont mis en évidence une augmentation de la qualité du modèle lorsque celui-ci prend en compte les retours utilisateurs. Ces résultats encourageants nous poussent à poursuivre nos travaux dans cette direction. Le système obtenu étant suffisamment mature, une perspective évidente est de le pousser en production, d'y intégrer progressivement les feedbacks utilisateurs et d'observer son évolution.

Un système en production doit faire face à une difficulté supplémentaire, introduite plus haut : l'utilisateur est susceptible, soit parce qu'il n'a pas compris, soit parce qu'il joue, de rentrer des phrases qui n'expriment aucun besoin en information. Il est alors essentiel, pour assurer la crédibilité du système, d'identifier correctement ces entrées afin de ne pas afficher la réponse à une question sans rapport. L'approche la plus simple est certainement d'entrainer un modèle de classification. Nous devrons probablement produire de nouveaux traits, plus pertinents pour cette tâche.

La substitution d'un composant par un autre entraîne toujours un risque de régression. L'identification et la mesure de ces régressions est un aspect critique lorsque l'on gère un système en production, d'autant plus lorsque, comme dans le scénario considéré, on veut remplacer un système de règles écrit à la main par un modèle issu d'un algorithme d'apprentissage automatique, difficile à ajuster une fois construit et souvent vu comme une boîte noire.

Les approches d'apprentissage actif telles que celle que nous présentons ici se marient en général très bien avec des algorithmes d'apprentissage en ligne (*online learning*), ceux-ci permettant d'intégrer progressivement de nouvelles données pour faire évoluer un modèle en temps réel. Dans les évaluations décrites plus haut, chaque modèle est à chaque fois construit intégralement à partir de l'ensemble des données d'apprentissage (en *batch*). Cela n'a posé aucun problème, étant donné la taille relativement réduite de notre corpus, mais nous voulons anticiper le passage à l'échelle en intégrant un algorithme d'apprentissage en ligne. Il faudra alors également penser à gérer le problème des traits représentant la popularité d'une question candidate, ceux-ci étant calculés sur l'ensemble du corpus.

Enfin, une intégration dans un champ de recherche avec une autocomplétion suggérant des questions proches veut épargner à l'utilisateur la nécessité de renseigner l'intégralité de sa question. Dans ce cas, la fonction de suggestion de questions doit être opérationnelle (même si de moins bonne qualité) dès les premiers caractères entrés par l'utilisateur. Cet aspect est déjà pris en compte par le composant d'analyse de questions, mais le réordonnancement des questions candidates, pour être pertinent, devra probablement être appris sur un corpus contenant des débuts de questions, que l'on pourrait construire en prenant les questions du corpus existant et en les tronquant à différentes tailles.

Références

ATTARDI, G., CISTERNINO, A., FORMICA, F., SIMI, M., TOMMASI, A., & ZAVATTARI, C. (2001). PiQASso: Pisa Question Answering System. In TREC.

BUNESCU, R., & HUANG, Y. (2010). LEARNING THE RELATIVE USEFULNESS OF QUESTIONS IN COMMUNITY QA. Conference on Empirical Methods in Natural Language Processing (pp. 97-107).

BURGES, C., SHAKED, T., RENSHAW, E., LAZIER, A., DEEDS, M., HAMILTON, N., & HULLENDER, G. (2005, August). Learning to rank using gradient descent. In Proceedings of the 22nd international conference on Machine learning (pp. 89-96). ACM.

CAO, Z., QIN, T., LIU, T. Y., TSAI, M. F., & LI, H. (2007, June). Learning to rank: from pairwise approach to listwise approach. In Proceedings of the 24th international conference on Machine learning (pp. 129-136). ACM.

CAO, Y., XU, J., LIU, T. Y., LI, H., HUANG, Y., & HON, H. W. (2006, August). Adapting ranking SVM to document retrieval. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 186-193). ACM.

CHAKRABARTI, S. (2004). Breaking through the syntax barrier: Searching with entities and relations. In Knowledge Discovery in Databases: PKDD 2004 (pp. 9-16). Springer Berlin Heidelberg.

CHAPELLE, O., METLZER, D., ZHANG, Y., & GRINSPAN, P. (2009, November). Expected reciprocal rank for graded relevance. In Proceedings of the 18th ACM conference on Information and knowledge management (pp. 621-630). ACM.

COSSOCK, D., & ZHANG, T. (2006, June). Subset ranking using regression. In Proceedings of the 19th annual conference on Learning Theory (pp. 605-619). Springer-Verlag.

CRAMMER, K., & SINGER, Y. PRANKING WITH RANKING. (2001). Advances in Neural Information Processing Systems, 14.

DAMLJANOVIĆ, D., AGATONOVIĆ, M., CUNNINGHAM, H., & BONTCHEVA, K. (2013). Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues. Web Semantics: Science, Services and Agents on the World Wide Web, 19, 1-21.

DEKLEVA, S. M. (1994). Is natural language querying practical?. ACM SIGMIS Database, 25(2).

DESERT, S. E. (1993). WESTLAW Is Natural v. Boolean searching: A performance study. J., 85, 713.

DITTENBACH, M., MERKL, D., & BERGER, H. (2003). A natural language query interface for tourism information. ENTER 2003: 10th International Conference on Information Technologies in Tourism.

FREUND, Y., IYER, R., SCHAPIRE, R. E., & SINGER, Y. (2003). An efficient boosting algorithm for combining preferences. The Journal of machine learning research, 4, 933-969.

HERBRICH, R., GRAEPEL, T., & OBERMAYER, K. (1999). Large margin rank boundaries for ordinal regression. Advances in neural information processing systems, 115-132.

HIRSCHMAN, L., & GAIZAUSKAS, R. (2001). Natural language question answering: the view from here. natural language engineering, 7(4), 275-300.

HOVY, E. H., GERBER, L., HERMJAKOB, U., JUNK, M., & LIN, C. Y. (2000, November). Question Answering in Webclopedia. In TREC (Vol. 52, pp. 53-56).

JÄRVELIN, K., & KEKÄLÄINEN, J. (2002). Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS), 20(4), 422-446.

KAUFMANN, E., & BERNSTEIN, A. (2010). Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. Web Semantics: Science, Services and Agents on the World Wide Web, 8(4), 377-393.

LAURENT, D., SEGUELA, P., & NEGRE, S. (2005). QRISTAL, système de Questions-Réponses. TALN & RECITAL, 1, 53-62.

MOLDOVAN, D. I., HARABAGIU, S. M., GIRJU, R., MORARESCU, P., LACATUSU, V. F., NOVISCHI, A., & BOLOHAN, O. (2002, November). LCC Tools for Question Answering. In TREC.

MOLDOVAN, D., HARABAGIU, S., PASCA, M., MIHALCEA, R., GOODRUM, R., GIRJU, R., & LASSO, V. R. (1999, November). A tool for surfing the answer net. In TREC-8 Proceedings.

LI, H. (2011). A short introduction to learning to rank. IEICE TRANSACTIONS on Information and Systems, 94(10), 1854-1862.

LI, H. (2014). Learning to rank for information retrieval and natural language processing. Synthesis Lectures on Human Language Technologies, 7(3), 1-121.

LI, P., WU, Q., & BURGES, C. J. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. In Advances in neural information processing systems (pp. 897-904).

LINCKELS, S., & MEINEL, C. (2005). A simple solution for an intelligent librarian system. In IADIS AC

LITKOWSKI, K. C. (2000, November). Syntactic Clues and Lexical Resources in Question-Answering. In TREC.

LIU, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

MIN, W., XIAOYU, Z., & MICHELLE, D. (2003). Question Answering By Pattern Matching, Web-Proofing, Semantic Form Proofing. In Proceedings of the TREC-12 Conference. Maryland: NIST Special Publication (pp. 165-169).

REICHERT, M., LINCKELS, S., MEINEL, C., & ENGEL, T. (2005, December). Student's Perception of a Semantic Search Engine. In CELDA (pp. 139-147).

SHASHUA, A., & LEVIN, A. (2002). Ranking with large margin principle: Two approaches. In Advances in neural information processing systems (pp. 937-944).

SRIHARI, R., & LI, W. (1999). Information extraction supported question answering. CYMFONY NET INC WILLIAMSVILLE NY.

SURDEANU, M., CIARAMITA, M., & ZARAGOZA, H. (2008). Learning to Rank Answers on Large Online QA Collections. In ACL (Vol. 8, pp. 719-727).

TAYLOR, M., GUIVER, J., ROBERTSON, S., & MINKA, T. (2008, February). Softrank: optimizing non-smooth rank metrics. In Proceedings of the 2008 International Conference on Web Search and Data Mining (pp. 77-86). ACM.

THOMPSON, C. W., PAZANDAK, P., & TENNANT, H. R. (2005). Talk to your semantic web. IEEE Internet Computing, 9(6), 75.

VOORHEES, E. M., & TICE, D. M. (1999, November). The TREC-8 Question Answering Track Evaluation. In TREC (Vol. 1999, p. 82).

WU, Q., BURGES, C. J., SVORE, K. M., & GAO, J. (2010). Adapting boosting for information retrieval measures. Information Retrieval, 13(3), 254-270.

XIA, F., LIU, T. Y., WANG, J., ZHANG, W., & LI, H. (2008, July). Listwise approach to learning to rank: theory and algorithm. In Proceedings of the 25th international conference on Machine learning (pp. 1192-1199). ACM.

XU, J., & LI, H. (2007, July). Adarank: a boosting algorithm for information retrieval. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 391-398). ACM.

YUE, Y., FINLEY, T., RADLINSKI, F., & JOACHIMS, T. (2007, July). A support vector method for optimizing average precision. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 271-278). ACM.

ZHENG, Z., ZHA, H., ZHANG, T., CHAPELLE, O., CHEN, K., & SUN, G. (2008). A general boosting method and its application to learning ranking functions for web search. In Advances in neural information processing systems (pp. 1697-1704).