

Chapitre 3

Apprentissage et Systèmes de Question-Réponse

3.1. Introduction

Les systèmes de Question-Réponse ont généralement une architecture modulaire, et la mise au point manuelle d'un certain nombre de leurs composants est difficile à réaliser. L'utilisation de méthodes d'apprentissage permet d'automatiser certaines parties du développement de ces systèmes : l'expert laisse à des algorithmes automatiques les travaux fastidieux de fouille dans des corpus ou d'optimisation de paramètres, et se concentre sur les problématiques de plus haut niveau, comme la représentation des données ou les spécifications fonctionnelles. Dans l'état de l'art, les méthodes d'apprentissage appliquées aux QR ont ainsi permis (1) d'augmenter les performances générales des composants de ces systèmes dans lesquelles ils sont appliqués, (2) de développer plus rapidement les composants ou le système complet, (3) de développer des systèmes qui peuvent être adaptés, sans intervention humaine, à plusieurs langues, ou encore (4) de réaliser des systèmes de QR utilisant très peu de ressources linguistiques ou de bases de connaissances.

Dans ce chapitre, nous allons présenter les principales approches utilisant l'apprentissage dans les systèmes de QR. Nous avons classé les techniques d'apprentissage employées dans ces systèmes suivant trois catégories : (a) celles à base d'extraction de chaînes de caractères fréquentes, (b) celles définies suivant le cadre de la classification supervisée et (c) celles issues du cadre de l'ordonnancement supervisé. Chacune de ces catégories correspond à un processus

2 Les systèmes de question-réponse

d'apprentissage générique ; autrement dit, à l'intérieur d'une même catégorie, les mêmes algorithmes d'apprentissage peuvent être utilisés. Par contre, en fonction des données qui sont utilisées durant l'apprentissage (ou plus précisément en fonction de la *représentation des données*), un même algorithme d'apprentissage peut être utilisé pour apprendre différents composants d'un système QR.

Dans les systèmes QR, les approches d'extraction de chaînes de caractères ont été utilisées pour identifier des morceaux de phrases qui permettent de localiser la réponse dans les documents. Plus précisément, pour chaque classe de question (i.e. pour une étiquette sémantique de la réponse), certains auteurs ont proposé de créer, automatiquement, une liste d'expressions qui apparaissent dans les passages contenant des mots-clefs de la question et la réponse. Ces expressions peuvent ensuite être utilisées pour générer des requêtes envoyées au moteur de recherche afin d'améliorer sa précision [AGI 01], ou encore comme patrons d'extraction [RAV 02]. Dans tous les cas, l'apprentissage nécessite en premier lieu une base de questions pour lesquelles la réponse est connue, une collection de documents (par exemple le Web), qui permet d'obtenir un grand nombre de passages qui contiennent la réponse, puis un algorithme permettant d'extraire, efficacement, les chaînes de caractères (ou expressions régulières) pertinentes. Les algorithmes d'apprentissage utilisés permettent donc d'éviter à des humains l'analyse manuelle de corpus textuels (qui doit être faite pour chaque classe de questions indépendamment), ce qui se traduit par un gain considérable en terme de temps de développement.

Le cadre de la classification supervisée, pour sa part, a été utilisé dans les systèmes QR pour apprendre soit le composant de classification de la question (i.e. le composant qui associe, à une question, l'étiquette sémantique de la réponse), soit le composant de sélection de la réponse (après l'extraction des réponses candidates). Par définition, un classifieur associe à un élément d'entrée une étiquette de classe ; la communauté d'apprentissage a développé de nombreux algorithmes permettant d'apprendre ce type de fonction à partir d'un ensemble d'exemples, c'est-à-dire à partir d'un ensemble d'éléments d'entrées pour lesquels l'étiquette de classe est connue. Ainsi, ce type d'algorithmes permet d'apprendre le composant de classification de la question à partir d'un ensemble de questions pour lesquelles les étiquettes sémantiques de la réponse sont connues. Pour sa part, le composant de sélection de la réponse est reformulé de la façon suivante : l'entrée est un couple (question, réponse candidate), pour lequel l'étiquette de classe est « + » si la réponse candidate est la bonne réponse à la question, et « - » sinon. Pour ces deux composants, l'intérêt de l'apprentissage est généralement important, en termes de temps de développement, mais aussi de performances [MET 05, KOL 06] ou de généralité : par exemple, [SOL 04] ont développé, grâce à l'apprentissage, un système de classification de questions qui peut être utilisé sans modification pour différentes langues.

Dans l'état de l'art de l'apprentissage pour les systèmes QR, les cadres d'extraction et de classification ont été les seuls cadres d'apprentissage utilisés. Nous décrivons cependant dans ce chapitre une troisième catégorie d'approches, plus originale, utilisant les algorithmes d'apprentissage de fonctions d'ordonnement. Ces fonctions prennent en entrée un ensemble, et renvoient une liste ordonnée de ses éléments. Par rapport au cadre de la classification, ce cadre d'apprentissage est particulièrement adapté aux QR dans les composants pour lesquels, en fonction de la question, un ensemble « d'alternatives » doivent être ordonnées entre elles. Nous appliquerons ainsi un algorithme d'apprentissage adapté pour ordonner les passages d'une collection dans l'étape de recherche d'information. Dans ce cas, l'apprentissage permet des gains de performance importants par rapport à un moteur de recherche classique, en particulier en permettant de combiner et de sélectionner un grand nombre de critères hétérogènes (syntaxiques, sémantiques et autres), avec une base d'apprentissage composée de questions pour lesquelles les documents contenant la réponse sont connus.

La suite de ce chapitre est organisée comme suit : dans la section 3.2., nous faisons un bref rappel des différents composants des systèmes QR, et précisons quels composants peuvent être appris et ce que l'apprentissage leur permet de gagner. Dans les sections 3.3 et 3.4. nous présentons les principales méthodes d'apprentissage qui ont été proposées dans l'état de l'art, en les regroupant selon les deux catégories décrites précédemment : extraction de chaînes de caractères ou classification supervisée. Dans la section 3.5., nous décrivons en détails l'approche d'apprentissage que nous avons proposée pour ordonner les passages (i.e. des paragraphes ou des ensembles de phrases) dans le module de recherche d'information : l'algorithme d'apprentissage, les représentations des passages, les données utilisées et les résultats obtenus. Enfin, la section 3.6., conclut le chapitre.

3.2. Systèmes de question-réponse et apprentissage

3.2.1. Architecture de référence des systèmes QR

L'apprentissage dans les systèmes QR est effectué au niveau de composants isolés. Afin de clarifier notre présentation, nous nous placerons dans le cadre d'une architecture standard (figure 1) des systèmes participant aux évaluations TREC pour le traitement des questions factuelles, dont les principaux composants sont :

- *Analyse de la question* (composant A. sur la figure 1) :

Ce module est généralement constitué de deux étapes : une première effectuant une analyse morphologique et syntaxique de la question, détermine les mots-clefs principaux (focus de la question [CHA 02], verbe principal, ...), et parfois les entités nommées apparaissant dans la question; une seconde étape détermine la *classe* de la question, c'est-à-dire l'étiquette sémantique de la réponse attendue.

4 Les systèmes de question-réponse

– *Génération d’une requête* (composant B.) :

A partir des mots-clefs identifiés dans l’analyse de la question, et parfois aussi de la classe de la question, ce composant génère une requête qui permet d’interroger un moteur de recherche.

– *Ordonner les passages de la collection susceptibles de contenir la réponse* (composant C.) :

A partir de la requête, et parfois des informations fournies par l’analyseur de questions, un certain nombre de passages de la collection sont sélectionnés afin d’être examinés par le composant d’extraction de la réponse (composant D., voir ci-dessous).

– *Extraction de la réponse* (composants D.) :

Ce composant extrait des passages retournés par le composant de recherche d’information (composant C.) des chaînes de caractères susceptibles d’être la réponse à la question. Ces chaînes de caractères sont appelées *réponses candidates* dans la suite de ce chapitre. Les traitements sous-jacents dépendent de la classe de la question, et il peut s’agir d’appliquer des patrons d’extractions (spécifique de cette classe), ou bien d’utiliser un système d’extraction d’entités nommées.

– *Sélection de la réponse* (Composant E.) :

Considérant l’ensemble des réponses candidates, ce composant détermine la réponse finale que le système doit renvoyer. Par rapport au composant d’extraction, l’étape de sélection de la réponse, qui se focalise généralement sur l’extraction de certains types de chaînes de caractères, utilise d’une part plus d’information sur la proximité lexicale ou sémantique entre le passage dans lequel les chaînes de caractères sont extraites et la question, et peut, d’autre part, utiliser des informations externes, comme le nombre de fois que la même chaîne de caractères a été extraite.

Il existe un grand nombre de variations de cette architecture de référence. Certains systèmes peuvent comporter plusieurs versions d’un même composant (par exemple plusieurs moteurs de recherche sur la même collection comme Google et Yahoo ! sur le Web), utiliser plusieurs sources d’information (collections fermées et Web, bases de connaissances externes comme le CIA FactBook), voire plusieurs sous-systèmes (comme [CHU 04] qui utilisent un système statistique à base d’apprentissage et un système à base de traitement de la langue). Bien que ces approches puissent augmenter les performances, elles rendent le développement des systèmes plus lourd : il faut d’une part adapter ou dupliquer les modules dont dépendent les composants concernés (par exemple, dans le cas de l’utilisation de plusieurs moteurs de recherche, il faut avoir un composant de génération de requête par moteur), et intégrer les sorties des différentes versions d’un même composant dans le composant suivant (par exemple, dans le cas de l’utilisation de plusieurs sources d’information pour l’extraction des réponses candidates, l’étape de sélection

de la réponse doit combiner, pour chaque réponse candidate, les sorties de chaque source).

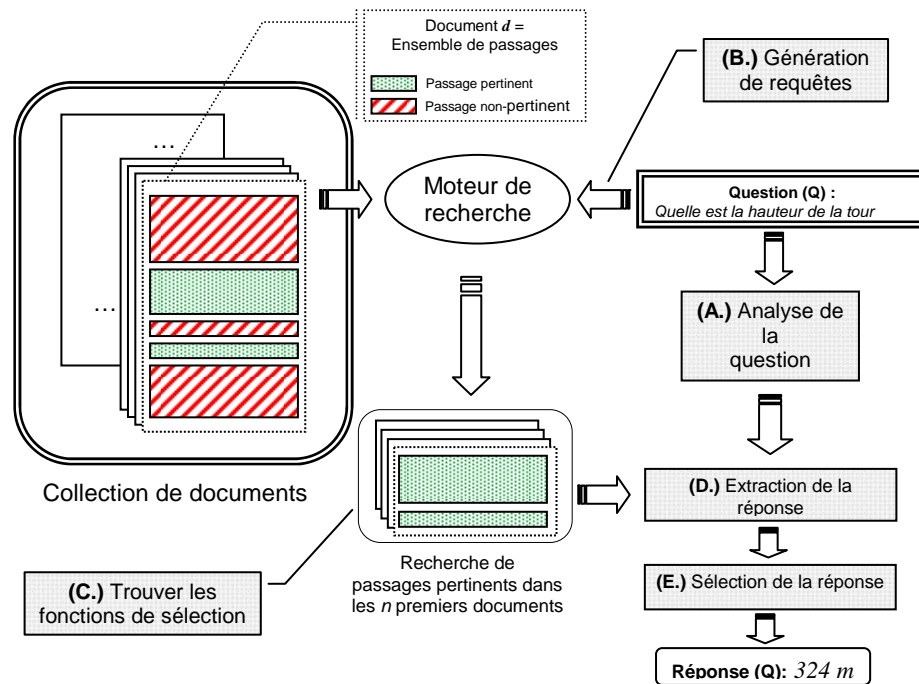


Figure 1. Schéma simplifié d'un système de QR. Les composants dans lesquels l'apprentissage peut être utilisé sont en gris.

3.2.2. Intérêt de l'apprentissage dans les différents composants

L'utilisation de l'apprentissage dans les systèmes QR va avoir comme motivation principale l'automatisation de nombreuses parties répétitives ou fastidieuses de la conception de ces systèmes. Dans cette section, nous passons en revue les composants qui peuvent bénéficier d'une telle automatisation. Les approches d'apprentissage à proprement parler seront étudiées plus en détails dans les sections 3.3., 3.4. et 3.5. Dans ce chapitre, nous ne décrivons l'apprentissage que dans les composants spécifiques aux QR. En particulier, l'ensemble des techniques d'apprentissage utilisées en traitement du langage ou en extraction d'information, qui permettent de créer des outils indépendamment des données de QR mais qui sont utilisées dans les systèmes QR ne seront pas étudiées ici. Cela inclut, par exemple, les approches d'apprentissage pour l'analyse syntaxique, l'étiquetage morphologique ou la reconnaissance d'entités nommées.

6 Les systèmes de question-réponse

Il existe deux principales spécifications des systèmes QR qui impliquent une répétitivité dans le développement de ces systèmes :

- traitement de plusieurs classes de questions (distance, lieu, personne, organisation, ...):

Chaque classe de questions oblige les concepteurs du système à créer de nouvelles règles permettant de reconnaître les questions de cette classe, et à adapter spécifiquement le composant d'extraction d'information : création de nouveaux patrons d'extraction adaptés ou d'un système permettant de reconnaître les entités nommées adaptées. Potentiellement, l'étape de génération de requêtes peut aussi être optimisée, ainsi que tous les composants qui prennent en entrée la classe de la question (par exemple la sélection de la réponse).

Ainsi, dans chacun de ces composants, des approches à base d'apprentissage, nécessitant uniquement un ensemble de questions pour lesquelles la réponse est connue permet d'acquérir une forte généralité par rapport aux classes de questions traitées. Il peut ainsi y avoir des gains importants en termes de temps de développement, mais aussi en termes d'évolutivité des systèmes.

- combinaison de plusieurs sources d'information ou d'un grand nombre de critères hétérogènes :

Une façon d'obtenir de bonnes performances est de donner aux différents composants un accès à un grand nombre d'informations différentes : par exemple, pour déterminer la classe d'une question, il peut être intéressant de regarder son pronom interrogatif, mais il n'est alors possible que d'avoir une faible granularité dans les étiquettes sémantiques possibles. Si, en plus du pronom interrogatif, la fonction de décision peut considérer les autres mots de la question, ainsi que leurs étiquettes morphologiques, ou encore l'arbre syntaxique, de plus grandes granularités peuvent être obtenues. Cependant, le nombre de critères qu'il est possible de considérer lorsque le développement est manuel est nécessairement faible, et créer une combinaison performante des critères va nécessiter des analyses poussées de corpus de questions. Ainsi, une méthode d'apprentissage qui permet d'être générique par rapport aux critères considérés (et à leur nombre) va permettre d'exploiter l'ensemble des informations disponibles, et donc d'atteindre des performances supérieures. Un second avantage de l'apprentissage lorsque l'on combine différents critères est la facilité avec laquelle il est possible d'ajouter/supprimer/tester de nouveaux critères et de nouvelles combinaisons. De ce fait, une fois encore, une procédure d'apprentissage adaptée permet des gains en termes de temps de développement importants.

Ces considérations sont aussi valables dans le cas où différentes sources d'information doivent être considérées. C'est le cas de l'ordonnement des passages avant l'extraction lorsque différents moteurs de recherche (ou différentes requêtes) sont utilisés, ou encore dans le composant de sélection de la réponse lorsque plusieurs corpus ou bases de connaissances sont utilisées.

Ainsi, l'apprentissage peut être intéressant pour une grande partie des composants, principalement pour gagner en généralité (et donc en temps de développement) et/ou en performances. La difficulté majeure est alors de formuler chaque composant comme un problème d'apprentissage et de collecter les données correspondantes. Les formulations des problèmes d'apprentissage correspondants aux différents composants peuvent être regroupées en trois catégories, qui dépendent non pas du composant considéré, mais de ce que les données permettent de déterminer automatiquement. La section 3.3., décrit ainsi des approches d'extraction de chaînes de caractères, utilisées dans les systèmes QR pour déterminer automatiquement des listes d'expressions ou des patrons d'extraction, d'une façon générique par rapport à l'ensemble des classes de questions considéré. Dans la section 3.4., nous présenterons les approches de classification, utilisées dans les composants de classification de la question ou de sélection de la réponse. Une fois encore, ces approches permettent de créer des composants génériques par rapport aux classes de questions considérées ou par rapport aux sources d'informations utilisées. Enfin, dans la section 3.5., nous présenterons une approche basée sur des algorithmes d'ordonnancement, qui peut être utilisée dans l'étape d'ordonnancement des passages avant l'extraction, lorsque plusieurs moteurs de recherche, plusieurs corpus de données ou de nombreux critères hétérogènes sont considérés.

3.3. Approches par extraction de chaînes de caractères

L'étude des systèmes QR a permis de montrer que l'analyse surfacique du texte était souvent suffisante pour localiser la réponse [SOU 02] : en fonction de la classe de questions, il existe un certain nombre de formulations qui sont souvent utilisées dans les passages contenant la réponse. Par exemple, pour une question de date de naissance (« Quelle est l'année de naissance de <PERSONNE> ? »), la chaîne de caractères « <PERSONNE> (<ANNEE> - » signifie souvent <ANNEE> est la réponse à la question. D'une façon identique, pour les questions de type définition, l'expression suivant le verbe « signifie » précédé du terme à définir est généralement la réponse à la question. Ainsi, un composant important des systèmes QR est le suivant : pour chaque classe de questions que le système peut traiter, il faut posséder une liste de mots (ou suite de mots) ou de chaînes de caractères qui permettent de localiser la réponse à la question.

La création de telles listes requiert l'étude d'un grand nombre de passages contenant la réponse à une question, pour un grand nombre de questions d'une même classe. Ce travail, de plus, doit être répété pour chaque classe de questions traitée par le système. Différents auteurs ont donc proposé ou utilisé des méthodes, indépendantes de la classe de question considérée, pour apprendre automatiquement ces listes. Ainsi, [AGI 01] apprennent des mots (ou des bi/tri-grams) apparaissant dans les passages contenant la réponse, ces mots étant ensuite ajoutés à une requête

8 Les systèmes de question-réponse

booléenne afin d'obtenir des passages plus pertinents en sortie du moteur de recherche (composant A. de la figure 1), et [RAV 02] ont proposé une méthode pour extraire automatiquement des chaînes de caractères pouvant servir de patrons d'extraction (utilisées alors dans le composant d'extraction, D. sur la figure 1). Cette dernière méthode a ensuite été utilisée dans de nombreux systèmes, comme ceux de [ECH 03] ou [PEN 05].

Techniquement, les approches d'extractions pour les QR sont issues inspirées du domaine de l'extraction d'information (voir par exemple [AMI 00]). Adaptées aux QR, elles se décomposent de la façon suivante : pour une classe de questions données, il faut collecter un ensemble de questions et de réponses associées. Pour chacune de ces questions, un ensemble de passages contenant la réponse est alors trouvé, généralement en utilisant internet interrogé avec une requête contenant un mot clé déterminant de la question (par exemple, la chaîne de caractères correspondante à <PERSONNE> dans une question de date de naissance comme décrit précédemment), ainsi que la chaîne de caractères correspondante à la réponse. Tous les n-grams [AGI 01] ou toutes les chaînes de caractères contenant à la fois le mot-clé de la question et la réponse [RAV 02] sont alors extraits des différents passages, puis un score est associé à chacun d'eux en fonction du recouvrement (proportion des passages, toutes questions comprises, pour lesquelles la réponse peut être trouvée grâce à la chaîne de caractères considérée) et de leur précision (proportion des passages dans lesquels la chaîne apparaît et permet effectivement de localiser la réponse). Les quelques chaînes de caractères ayant les meilleurs scores sont ensuite conservées. Ce processus est ensuite répété pour chaque classe de questions.

L'implémentation effective du stockage de l'ensemble des chaînes de caractères et des méthodes de score ne seront pas discutées ici, le lecteur intéressé peut se référer à [AGI 01, RAV 02]. Ce qu'il est important de noter est, (1) la généralité de l'approche par rapport aux classes de questions, (2) les grandes quantités de texte sur laquelle la méthode peut effectuer des statistiques (cela permet, en particulier, d'obtenir un plus grand nombre de chaînes de caractères pertinentes et donc d'augmenter le recouvrement de la liste de patrons d'extraction [PEN 05]). Par rapport aux méthodes manuelles, il est ainsi possible d'obtenir des gains de performance important tout en gagnant en termes de temps de développement. La difficulté de la mise en œuvre de ces approches réside, cependant, dans le fait que les questions qu'il est possible de traiter doivent être syntaxiquement simples (composée d'une seule expression-clef facilement identifiable), mais aussi dans l'établissement des heuristiques qui allouent des scores aux chaînes de caractères, qui doivent être définies selon le composant considéré et pour lesquelles il n'existe aucun cadre d'étude général.

3.4. Approches de classification supervisée

Les approches de l'état de l'art utilisant l'apprentissage qui ont eu le plus de succès sont celles qui se basent sur le cadre d'apprentissage de la classification. Ces approches ont été proposées pour le composant de classification de la question (composant A. sur la figure 1), ou le composant de sélection de la réponse (composant E.). Le cadre général de la classification supervisée, qui consiste à apprendre des fonctions qui ont comme espace de sortie un ensemble discret (généralement contenant peu d'éléments), a été profondément étudié en apprentissage ; il existe donc un certain nombre d'algorithmes extrêmement performants et faciles à mettre en place.

La majorité des approches utilisent le cadre de la classification de vecteurs : pour le composant de classification de la question, le concepteur du système extrait un certain nombre de caractéristiques des questions, qui servent ensuite à former un vecteur représentatif. Chacun des composants est alors formulé comme étant un classifieur de vecteurs :

- pour le composant de classification de la question, la base d'apprentissage est alors constituée de (vecteurs représentatifs de) questions, ainsi que de leur classe associée. Un algorithme d'apprentissage de classifieurs de vecteurs peut alors être appliqué pour trouver la correspondance entre les vecteurs représentatifs et les classes.

- pour le composant de sélection de la réponse, chaque réponse candidate fournie par l'extracteur de réponses (composant D. sur la figure 1) est représentée par un vecteur dans lequel chaque caractéristique prend en compte à la fois cette réponse candidate et la question pour laquelle elle a été extraite. La classe associée à une réponse candidate est alors binaire : « 1 » si la réponse candidate est la bonne réponse à la question, et « -1 » sinon. A partir d'un ensemble de questions pour lesquelles la réponse est connue, une base d'apprentissage est alors constituée de l'union, pour l'ensemble des questions, des (vecteurs représentatifs des) réponses candidates et de leurs classes. Un algorithme de classification binaire est alors utilisé pour établir la correspondance entre les représentations des réponses candidates et le fait qu'elles sont (ou non) les bonnes réponses.

Dans la suite de cette section, nous commençons par une brève description formelle du problème de classification en apprentissage. Nous présentons ensuite, pour chacun des deux composants concernés, les caractéristiques utilisées pour représenter les exemples en vecteurs, ainsi que l'intérêt et les performances des approches d'apprentissage.

3.4.1. Introduction à la classification

La classification de vecteurs est une tâche de l'apprentissage supervisé où il s'agit de construire une fonction de prédiction f à valeurs discrètes couramment appelé *classifieur*, dont l'espace d'entrée est $\mathcal{X} \subseteq \mathbb{R}^d$ et l'espace de sortie est $\mathcal{Y} = \{1, \dots, c\}$. Un algorithme d'apprentissage permet de trouver automatiquement un classifieur à partir d'un ensemble d'apprentissage constitué de couples $(x_i, y_i)_{1 \leq i \leq m}$ avec $x_i \in \mathcal{X}$ et $y_i \in \mathcal{Y}$, et où y_i correspond à la sortie (ou classe) que la fonction à apprendre f doit associer au vecteur d'entrée x . Une fois apprise, cette fonction peut être ensuite utilisée pour prédire la classe $k \in \mathcal{Y}$ qui correspond à un vecteur d'entrée quelconque $x \in \mathcal{X}$.

Une des techniques les plus répandues dans les composants type *classification* des systèmes de QR est l'algorithme des Machines à Vecteurs Support (MVS). Cet algorithme, introduit par V. Vapnik (voir [BUR 98] ou [VAP 00] pour des descriptions détaillées) permettant d'apprendre des classifieurs binaire, qui sont des combinaisons linéaires de caractéristiques dans un espace vectoriel. Le cadre formel des MVS est extrêmement général, et par le biais d'une théorie puissante des fonctions noyaux, l'algorithme peut être utilisé pour apprendre des fonctions non-linéaire, voire des classifieurs sur des espaces d'arbres (voir [BUR 98] ou [VAP 00] pour des descriptions générales des fonctions noyau, et [COL 02] pour les noyaux permettant de classer des arbres).

Un grand nombre d'implémentations des MVS dans le cas général (utilisant des noyaux pour la classification non-linéaire, acceptant des données bruitées et dans le cas multiclassés) existent. L'implémentation la plus utilisée dans les systèmes QR est LIBSVM [CHA 07].

3.4.2. Classification de la question (composant A.)

Le composant de classification de la question s'écrit naturellement comme un problème d'apprentissage de classifieur multiclassés : à partir d'une représentation (généralement vectorielle) des questions, l'ensemble de sortie du classifieur est l'ensemble des étiquettes sémantiques acceptées par le système. L'intérêt de l'apprentissage pour créer le composant de classification est double : d'une part, il permet au composant d'être facilement évolutif par rapport à l'ensemble des étiquettes sémantiques acceptées ; d'autre part, il permet d'utiliser un grand nombre de caractéristiques et donc d'obtenir des classifieurs ayant d'excellentes performances même en traitant un grand nombre de classes.

L'ensemble des caractéristiques des questions qui ont été proposées dans l'état de l'art pour apprendre des classifieurs peuvent être regroupées en 9 catégories :

- mots apparaissant dans la question [HAC 03] [LIR 02] [ZHA 03] [MET 05],

- bi-grams, tri-grams [HAC 03][LIH 05] [ZHA 03] [MET 05],
- entités nommées [HAC03], [LIR 02],
- étiquettes morphologiques des mots [LIR 02, HAC 03, [MET 05],
- *chunks* (expressions sans chevauchement), mot principal du *chunk* [LIR 02],
- *synsets* de WordNet [LIH 05],
- mots sémantiquement reliés à une classe de questions :

[LIR 02] associent à chaque classe de questions une liste de mots qui tendent à apparaître souvent dans des questions de cette classe ; [MET 05] utilisent des hyperonymes (issus de WordNet) du mot principal de la question, car certains hyperonymes tendent à apparaître souvent dans les questions de certaines classes données.

Bien que des méthodes d'extraction de chaînes de caractères similaires à celles décrites section 3.3., puissent être utilisées pour créer des listes comme celles de [LIR 02], cet apprentissage n'a pas été effectué dans l'état de l'art,

- arbres syntaxiques [ZHA 03],
- dépendances syntaxiques [LIH 05].

L'apprentissage et l'évaluation des classifieurs de questions sont généralement réalisés sur le corpus créé par [LIR 02], constitué de 5,500 questions d'apprentissage, le test étant réalisé sur les 500 questions de la collection TREC 10. Dans cette base, les étiquettes sémantiques (i.e. les classes de question) sont organisées hiérarchiquement, en deux niveaux : un premier niveau contient 6 classes génériques (par exemple lieu, humain, etc.), chacune divisée en un certain nombre de sous-classes (50 sous-classes au total), et, à chaque question de la base est associée une seule étiquette sémantique.

Au niveau des performances, des résultats rapportés par les auteurs de [HAC 03] indiquent que les MVS avec un noyau linéaire ou gaussien sont marginalement plus performants qu'un certain nombre d'autres algorithmes d'apprentissage. D'une façon plus générale, les différences significatives de performances proviennent des jeux de caractéristiques utilisées : lorsque le classifieur est entraîné pour reconnaître uniquement les 6 classes génériques, les performances en test sont de l'ordre de 85% de bonne classification en utilisant uniquement les mots contenus dans la question [HAC 03, ZHA 02, LIH 05, MET 05, LIR 02, ZHA 03], et de 90% en utilisant uniquement l'arbre syntaxique [ZHA 03] avec un SVM utilisant un noyau d'arbres [COL 02]. A partir de la représentation sac-de-mots, des gains marginaux peuvent être apportés par l'utilisation des bi-grams, des entités nommées ou des étiquettes morphologiques (de l'ordre de 1-2% chaque type de caractéristique). En général, l'apport de chacune de ces caractéristiques est non significatif, mais l'utilisation de l'ensemble d'elles permet des gains significatifs [MET 05, LIR 02, ZHA 02]. Enfin, le type de caractéristiques le plus performant, outre les mots de la question, utilise

12 Les systèmes de question-réponse

les mots sémantiquement reliés, pour lesquels la performance en classification augmente significativement, d'à peu près 5% [LIR 02, MET 05]. Les performances en classification dans le cas où le classifieur est entraîné sur les 50 classes sont de l'ordre de 5% inférieures par rapport au cas à 6 classes, mais les gains relatifs apportés par les différents types de caractéristiques sont identiques.

L'apprentissage a donc été utilisé pour produire des classifieurs de questions performants. Par ailleurs, la performance élevée des classifieurs utilisant uniquement mots ou des n -grams ont permis de créer des composants qui possèdent aussi une généralité par rapport à la langue utilisée : à partir de ce type de caractéristiques, [SOL 04] ont créé un classifieur de questions applicable à plusieurs langues européennes sans modification, et [PUR 07] ont pu développer un système pour l'indonésien, malgré le peu de ressources linguistiques disponibles pour cette langue.

3.4.3. Sélection de la réponse (composant E.)

L'apprentissage a été fortement utilisé dans le composant de sélection de la réponse. Les principales motivations de l'apprentissage dans ce composant sont (1) la nécessité de certains systèmes de combiner plusieurs sources d'informations ou plusieurs extracteurs de réponse, (2) la nécessité de ce composant de pouvoir être modifié facilement (il se situe en effet en dernière position de la chaîne de traitement, donc doit potentiellement être adapté dès qu'un des autres composants est modifié), et enfin (3) la nécessité, pour la plupart des systèmes de combiner des informations issues de l'extracteur de réponse (des scores de confiance, par exemple), ainsi que des informations sur la question initiale et le passage duquel a été extrait la réponse.

Comme nous l'avons dit au début de la section 3.4., le composant de sélection de la réponse peut être considéré comme un classifieur binaire de réponses candidates : la classe « 1 » est associée aux bonnes réponses, et « -1 » aux mauvaises. Il est ensuite possible d'apprendre un classifieur (par exemple une MVS) sur une base d'apprentissage, à partir d'une représentation vectorielle des réponses candidates. Lorsque les paramètres de la MVS sont appris, la sélection de la réponse à renvoyer par le système parmi les candidates se fait comme suit : pour chaque réponse candidate, la sortie du classifieur est considérée comme un score de confiance sur la pertinence de la réponse par rapport à la question. La réponse candidate obtenant le meilleur score est alors renvoyée par le système.

L'ensemble des caractéristiques permettant d'obtenir les représentations vectorielles des réponses candidates varient beaucoup selon les systèmes, mais elles peuvent être regroupées en 5 catégories :

– *correspondance entre le contexte d'apparition de la réponse candidate et le contexte de la question* :

Ici, le terme « contexte » peut avoir différents sens : il peut s'agir du nombre de mots communs entre la question et l'ensemble des mots qui apparaissent avant ou après la réponse candidate dans le passage duquel elle est extraite [ITT 01, RAV 03, PER 01, NGR 01]. Ce type de comptage peut être raffiné, en considérant séparément différents types de mots de la question (entités nommées, focus, etc.). Un autre type d'approche [JIJ 04, KOL 06] considère comme « contexte » d'une réponse candidate les cooccurrences entre des mots (ou des n-grams) de la question et la réponse candidate sur le Web [JIJ 04, KOL 06].

– *utilisation de bases de connaissances externes* :

Certains systèmes [JIJ 04, MIL 04, AND 04, KOL 06] ont des moyens d'interroger des bases de connaissances externes afin de vérifier si la base de connaissance connaît la réponse à la question, ou bien si la candidate est du type de réponse attendu (la base de connaissance sert alors à identifier le type sémantique de la réponse candidate). [ITT 01] utilise d'une façon similaire les définitions de WordNet [FEL 98] comme base de connaissance pour les questions de définition.

– *type sémantique de la réponse candidate/patrons d'extractions* :

Utilisé par tous les systèmes (sauf ceux pour lesquels l'extracteur de réponse filtre préalablement les candidates selon leur type), ces caractéristiques mesurent l'adéquation entre le type sémantique de la réponse candidate et celui de la question, ou bien la confiance dans le patron qui a extrait la réponse. Des bases de connaissances peuvent être utilisées, WordNet, ou encore les sorties des systèmes d'extraction nommées.

– *caractéristiques syntaxiques* :

Variante selon le type de question, [ITT 01] utilise par exemple ce type de caractéristiques pour les questions de définition : il définit des caractéristiques binaires si, par exemple, la réponse candidate est le sujet du verbe « to be » dans une phrase, et que le verbe est lui-même suivi de tous les mots-clefs de la question. [SUZ 02] utilisent eux aussi des caractéristiques basées sur la syntaxe des paragraphes desquels sont extraits les réponses.

– *redondance de la candidate extraite* :

[PER 01], et surtout [RAV 03] utilisent le nombre de fois que la même réponse candidate a été extraite de différents passages pour la même question.

A partir de ces caractéristiques et de la méthode pour générer l'ensemble d'apprentissage décrite au début de la section 3.4., des algorithmes habituels peuvent être utilisés, comme des MVS [SUZ 02, MIL 04, AND 04], des arbres de décision [NGR 01, PER 01], ou encore des modèles probabilistes [ITT 01, JIJ 04, MIL 04]. En général, l'apprentissage est implanté au cœur du système, donc il n'existe pas de comparaison entre combinaison avec ou sans apprentissage, mais les systèmes de

[ITT 01, NGR 01, PER 01], étaient au niveau de l'état de l'art (donc au niveau de performance des systèmes n'utilisant pas l'apprentissage d'une façon générale). Seuls [JIJ 04] et [KOL 06] testent des méthodes de combinaison plus naïves (par exemple une simple somme non pondérée). Les gains de performance relatifs (en terme de bonne sélection de la réponse) apportés par l'apprentissage sont généralement de quelques pourcents (mais significatifs).

Pour finir sur le module de sélection de la réponse, certains auteurs notent que le cadre de la classification, tel qu'il est décrit précédemment, n'est pas forcément pertinent pour apprendre des fonctions de sélection [JIJ 04, RAV 03]. Ce point sera discuté dans la section suivante, avec comme réponse le cadre d'apprentissage de fonctions d'ordonnement. [RAV 03] rapporte des gains de performances très significatifs en utilisant, à la place de la classification, une méthode d'apprentissage proche de celle que nous allons présenter.

3.5. Approche utilisant l'ordonnement supervisé

Un troisième cadre d'apprentissage qui peut être utilisée dans les systèmes QR est celui de l'apprentissage de fonctions d'ordonnement. Introduite en 1998 et résumée dans [FRE 03, USU 06], cette problématique correspond à apprendre des fonctions qui ordonnent des éléments entre eux, plutôt que de les classer. Elle est, en particulier, très adaptée aux problèmes de recherche d'information, où les documents d'une collection doivent être ordonnés entre eux par rapport à une requête utilisateur. Dans ce cas, il s'agit de calculer un score de pertinence (par rapport à la requête) pour chaque document de la collection, puis d'ordonner les documents par ordre de scores croissants. L'objectif est alors que les documents pertinents pour la requête obtiennent des meilleurs scores par rapport aux documents non pertinents.

Bien que le cadre de la classification puisse être appliqué en recherche d'information (en apprenant à classer les documents en deux classes : « 1 » si le document est pertinent pour la requête, « -1 » sinon), ce dernier paraît peu adapté pour cette tâche. En effet, dans le cas d'erreurs de classification (par exemple un document pertinent classé « -1 » par le classifieur) on ne peut pas savoir où est positionné cet élément pertinent dans la liste ordonnée créée par le classifieur : la seule information disponible est qu'il est en dessous de tous les documents classés « 1 ». Il peut donc être à la fin de la liste aussi bien qu'au milieu. Lorsque l'on apprend des fonctions qui ont pour but d'ordonner des éléments avec des classes bruitées (comme c'est le cas en recherche d'information), cette absence d'information peut empêcher d'apprendre des fonctions performantes. L'avantage du cadre de l'ordonnement est que (1) la fonction apprise a des valeurs réelles, c'est donc une vraie fonction de score, et (2) la fonction est apprise en *comparant* les *scores relatifs* entre différents éléments. Au contraire, en classification, la fonction

alloue à chaque élément une étiquette de classe, et le critère ne prend pas en compte les prédictions faites pour les autres éléments

Dans cette section, nous décrivons l'approche d'apprentissage que nous avons utilisée pour apprendre la fonction qui ordonne les passages (composant C. sur la figure 1) avant le composant d'extraction de la réponse (composant D. figure 1). Elle est basée sur un algorithme original, inspiré de l'algorithme RANKBOOST [FRE 03], et permet (1) d'obtenir de bonnes performances par rapport à un moteur de recherche habituel, et (2) de combiner un grand nombre de critères hétérogènes des passages textuels, par exemple des critères de proximité des mots de la question dans le passage, ou plus généralement des critères syntaxiques ou sémantiques, utilisant la ressource linguistique WordNet [FEL 98].

Suivant l'état de l'art des méthodes de recherche de passages dans les systèmes QR de [TEL 03], notre approche pour renvoyer des passages susceptibles de contenir la réponse à une question donnée est divisée en deux étapes : (1) un moteur de recherche habituel est d'abord interrogé pour renvoyer un grand nombre de passages contenant la réponse, et (2) ces passages sont ensuite réordonnés par une fonction de score spécifique des QR, prenant en compte des critères syntaxiques ou linguistiques qui ne sont pas considérés par les moteurs de recherche habituels. Par rapport aux travaux décrits dans [TEL 03], l'originalité de notre approche est double. D'une part, nous utilisons l'apprentissage pour combiner les différents critères utilisés dans l'étape (2). D'autre part, utilisant la généralité de l'apprentissage par rapport au nombre de caractéristiques considérées, nous pouvons considérer un grand nombre de caractéristiques potentiellement intéressantes. Pour cela, nous utilisons une méthode de génération automatique de caractéristiques de passages. En termes de temps de développement, nous évitons donc l'étude approfondie de chaque caractéristique qui est nécessaire dans les méthodes de combinaisons manuelles décrites dans [TEL 03]. Enfin, notre approche permet de gagner en généralité par rapport aux méthodes manuelles. En effet, il est maintenant très facile d'introduire de nouvelles caractéristiques pour étudier leur impact sur les performances : il suffit de ré-entraîner le système, ce qui ne nécessite aucune intervention humaine.

Dans la suite de cette section, nous commençons par décrire plus précisément les caractéristiques des passages que nous avons utilisées, ainsi que notre méthode automatique de génération de caractéristiques (section 3.5.1.). La section 3.5.2., présente l'algorithme d'ordonnement que nous avons utilisé, et la section 3.5.3., décrit le protocole expérimental et les résultats.

3.5.1. Caractéristiques utilisées

Afin de pouvoir utiliser un système d'apprentissage qui ordonne des passages selon qu'ils contiennent ou non la réponse à la question courante, il est nécessaire de posséder des caractéristiques pertinentes des passages. La majorité des caractéristiques que nous allons utiliser sont générées automatiquement par une technique originale. Cette méthode de génération est fondée sur des *règles de formulation de requêtes* (RFR). Une RFR est une règle qui génère une requête (i.e. un ensemble de mots-clefs) étant donné une question. Nous les représentons comme des listes de doublets [(*règle d'extension de mot*, *catégorie de mots*)], avec les définitions suivantes :

- une *règle d'extension de mot* R_{EM} est une fonction qui prend en entrée un mot ou un ensemble de mots, et qui utilise un lien particulier L de l'outil linguistique WordNet (il peut s'agir par exemple du lien « hyperonyme », « synonyme » ou encore « hyponymes »). Si l'entrée est composée d'un mot M , la sortie est l'ensemble des mots qui sont reliés, dans WordNet, à M par le lien L . Si l'entrée est constituée d'une liste de mots $[M_1, \dots, M_k]$, la fonction renvoie l'union des ensembles $R_{EM}(M_1)$ et $R_{EM}([M_2, \dots, M_k])$.

- une *catégorie de mots* est un ensemble de mots apparaissant dans la question courante, qui vérifie une contrainte syntaxique ou morpho-syntaxique donnée. Dans nos travaux, nous avons utilisé l'analyseur de questions du système QALC [CHA 02] afin d'obtenir ces catégories. Des exemples de catégories de mots sont « noms », « noms propres », « focus », « verbe principal », etc.

Ainsi, par exemple, pour une question donnée, la RFR [(*identité*, *focus*), (*synonymes*, *verbe principal*)] renvoie une requête composée de tous les mots du focus de la question et de tous les synonymes du verbe principal présents dans WordNet. Sur la base de toutes les combinaisons possibles entre les *règles d'extension de mots* et les *catégories de mots*, nous avons sélectionné 112 RFR avec une méthode heuristique [USU 04]. A partir de ces RFR, nous avons obtenu 112 caractéristiques des passages grâce à l'association suivante:

A chaque RFR $\mathcal{R} : \{questions\} \rightarrow \{requetes\}$ est associée une fonction caractéristique $f_{\mathcal{R}} : \{questions\} \times \{passages\} \rightarrow IR \cup \{\perp\}$ telle que $f_{\mathcal{R}}(q, p) = \mathbf{score}(\mathcal{R}(q), p)$ si p est dans les N premiers passages renvoyés par le moteur de recherche interrogé avec la requête $\mathcal{R}(q)$, et \perp sinon. Ici, \perp signifie que $f_{\mathcal{R}}$ n'alloue aucun score au passage p . La fonction de **score** que nous avons utilisé dans nos expériences est la mesure cosinus du moteur de recherche MG [WIT 99].

A cet ensemble de 112 caractéristiques, nous avons ajouté 5 autres caractéristiques inspirées du système QR d'IBM [ITT 00] et évaluées comme étant parmi les plus performantes dans l'étude comparative de [TEL 03].

3.5.2. Boosting pour l'ordonnement de passages

L'algorithme de Boosting constitue une des idées clés des années 90 en apprentissage. Cet algorithme a été au départ introduit pour la tâche de classification [SCH 90] et il a été depuis, étendu à d'autres tâches d'apprentissage. La motivation initiale de cet algorithme est de trouver itérativement un classifieur performant en combinant itérativement les sorties de plusieurs classifieurs *peu performants*.

Un des algorithmes de Boosting le plus populaire est probablement l'algorithme d'ADABOOST (*ADaptive BOOSTing*) [FRE 96]. Cet algorithme prend en entrée, en plus d'un ensemble d'apprentissage, un second algorithme de classification. A chaque itération, ADABOOST appelle le second algorithme afin d'apprendre un nouveau classifieur qui corrige les erreurs de prédiction de la combinaison linéaire courante. Une fois ce nouveau classifieur appris, il est ajouté dans la combinaison linéaire, avec un poids associé qui maximise la performance en apprentissage de la nouvelle combinaison. L'idée centrale de cet algorithme est de maintenir, à chaque itération t une distribution de probabilité D_t (ou un ensemble de poids). Le $i^{\text{ème}}$ classifieur à ajouter à la combinaison c_t , est alors déterminé par la sortie du second algorithme de classification, qui utilise comme base d'apprentissage la base initiale des exemples, pondérés par D_t ; l'erreur empirique sur une base d'apprentissage S de taille m qu'il minimise est la suivante:

$$R_m(c_t, S) = \sum_{i=1}^m D_t(i) [[c_t(x_i) \neq y_i]]$$

Où $[[\pi]]$ vaut 1 si le prédicat π est vrai et 0 sinon. R_m est donc la moyenne pondérée des erreurs du classifieur final c_t sur la base S . Initialement la distribution D_t est uniforme sur les exemples d'apprentissage, mais à chaque itération, les poids des exemples mal classés par la combinaison courante sont augmentés de façon à ce que le classifieur d'itération suivante se focalise sur ces exemples. Ainsi, le $i^{\text{ème}}$ classifieur corrige les erreurs de la combinaison courante.

Le poids associé au classifieur courant est ensuite déterminé pour minimiser l'erreur de la nouvelle combinaison. La distribution D_t est alors mise à jour pour associer un poids plus fort aux exemples mal classés par cette nouvelle combinaison.

Récemment, [FRE 03] ont proposé une adaptation de l'algorithme ADABOOST, appelé RANKBOOST, à la tâche d'ordonnement. RANKBOOST est basé sur l'algorithme ADABOOST suivant le principe de réduction de l'ordonnement à la classification [USU 06]: il construit itérativement une combinaison linéaire de fonctions de base, en adaptant à chaque itération une distribution de probabilité sur

l'ensemble des paires composées d'exemples (pertinent, non-pertinent)¹. Dans notre cas de recherche de passages dans les systèmes de QR, un exemple pertinent est un passage contenant la réponse à la question pour laquelle il a été renvoyé, et un exemple non-pertinent est un passage ne contenant pas la réponse. L'interprétation de la distribution de probabilité est la même que dans le cas de la classification: plus le poids assigné à une paire est élevé, plus la combinaison courante inverse l'ordre. Comme pour le cas de classification, l'algorithme RANKBOOST pondère itérativement les paires d'exemples et entraîne un algorithme de base suivant cette distribution.

A part la distribution définie sur les paires d'exemples, il y a une autre différence avec la description générale d'ADABOOST (qui est due à des problèmes de complexité d'apprentissage): chaque algorithme de base est une simple fonction de décision binaire associée à chaque caractéristique. Au lieu d'apprendre un classifieur *peu performant* à chaque itération, comme dans ADABOOST, cet algorithme sélectionne simplement une simple règle de décision basée sur une des caractéristiques, suivant la distribution courante.

3.5.2.1. Sélection des règles de décision binaires

La sortie finale de l'algorithme est une somme pondérée de ces règles de base. [FRE 03] proposent d'utiliser des règles qui assignent des scores appartenant à $\{0, 1\}$ exemples. Chaque règle h est automatiquement dérivée d'une caractéristique donnée en seuillant sa valeur par rapport à une valeur donnée. Ainsi, étant donnée une fonction de préférence f et un seuil θ , h est définie comme:

$$h(p) = \begin{cases} 1, & \text{si } f(p) \geq \theta \\ 0, & \text{si } f(p) < \theta \\ nd, & \text{si } f(p) = \perp \end{cases}$$

Où $nd \in \{0, 1\}$. La règle assigne ainsi le score par défaut nd aux passages p qui ne sont pas scorés par la caractéristique (e.g. $f(p) = \perp$). Concrètement, à chaque caractéristique est attribué un ensemble fixe N_θ de seuils θ .

L'utilisation des règles simples comme celles décrites plus haut assure que la complexité de l'algorithme reste linéaire en nombre de règles de base tandis qu'avec les caractéristiques initiales, la complexité de cet algorithme sera linéaire au produit du nombre des caractéristiques et des passages $N_{passages}$.

On remarque que le nombre de règles binaires maximal qu'on peut dériver est $2 * N_q * N_{caractéristiques}$ où $N_{caractéristiques}$ est le nombre de caractéristiques (117 dans notre cas) et que $N_{caractéristiques} \ll N_{passages}$.

¹Appelées *paires cruciales* dans la suite.

3.5.2.2. Adaptation de l'algorithme à l'ordonnement de passages

Dans sa version originale, l'algorithme de RANKBOOST était proposé pour le cas d'ordonnement appelé *ordonnement d'instances* (voir par exemple [USU 06]) qui, dans notre cas, ne permet d'apprendre qu'une fonction qui ordonne les passages pour une question donnée. Cependant, dans notre cas, nous souhaitons avoir une seule fonction qui ordonne les passages pour toutes les questions. Une légère modification est alors nécessaire.

Formellement, soit Q l'ensemble des questions dans la base d'apprentissage et P_q^1 et P_q^0 les ensemble respectifs de passages contenant ou pas la réponse à la question q . A chaque itération t , l'approche proposée maintient une distribution D_t sur les paires de passages dans $P = \bigcup_q P_q^1 \times P_q^0$. Un poids élevé affecté à une paire

indique que la fonction d'ordonnement courante a tendance d'ordonner un passage contenant la réponse à une question donnée au-dessous d'un passage ne la contenant pas. Comme pour ADABOOST, l'algorithme commence avec une distribution uniforme sur l'ensemble des paires dans P . A chaque itération t , l'algorithme de base sélectionne une règle h_t ainsi que son poids associé α_t . La règle h_t choisie est celle qui minimise la probabilité de mal-ordonnement définie grâce à D_t . La distribution D_t est mise à jour grâce à h_t et le poids α_t pour toutes les paires $(p_q^0, p_q^1) \in P$ d'après la règle suivante:

$$D_{t+1}(p_q^0, p_q^1) = \frac{D_t(p_q^0, p_q^1) \exp(\alpha_t (h_t(p_q^0) - h_t(p_q^1)))}{Z_t}$$

Où Z_t est le facteur de normalisation choisi de façon à ce que D_{t+1} reste une distribution à l'itération suivante :

$$D_{t+1}(p_q^0, p_q^1) = \sum_q \sum_{p_q^0, p_q^1} D_t(p_q^0, p_q^1) \exp(\alpha_t (h_t(p_q^0) - h_t(p_q^1))) \quad (1)$$

Pour un poids $\alpha_t > 0$ et une question q , si la règle h_t ordonne le passage pertinent p_q^1 au-dessous d'un passage non-pertinent p_q^0 (i.e. $h_t(p_q^0) - h_t(p_q^1) = 1$), la valeur de la distribution $D_{t+1}(p_q^0, p_q^1)$ croît et elle décroît dans le cas contraire.

Après apprentissage la fonction de score finale est la combinaison linéaire des règles de base : $H = \sum_t \alpha_t h_t$.

Le but de l'algorithme est de minimiser l'erreur d'ordonnement moyenne de la fonction de score finale H sur la base d'apprentissage :

$$Rloss = \sum_q \sum_{p_q^0, p_q^1} D(p_q^0, p_q^1) [[H(p_q^0) \geq H(p_q^1)]]$$

Où D est la distribution initiale sur les paires cruciales pour toutes les questions.

Il est facile de voir que $Rloss$ est majorée par $\prod_t Z_t$ ([USU 06]). A chaque itération

h_t et α_t sont alors choisis de façon à ce que $Z_t \leq 1$. Ce choix garantit alors que la borne supérieure de $Rloss$ décroît à chaque itération. L'optimisation de $Rloss$ est alors faite en minimisant sa borne supérieure ou d'une manière équivalente en minimisant Z_t à chaque itération.

On remarque que la complexité de la mise à jour de D_t est linéaire au produit du nombre de questions et de paires *cruciales* de la base d'apprentissage (équation 1). Pour diminuer cette complexité, nous avons suivi l'idée centrale de l'algorithme RANKBOOST qui consiste à maintenir une distribution sur les questions a_t^q et une distribution sur les exemples v_t^q (et non sur les paires d'exemples) et à réécrire la distribution D_t sous la forme :

$$\forall q, \forall (p_q^0, p_q^1) \in P_q^0 \times P_q^1, D_t(p_q^0, p_q^1) = a_t^q v_t^q(p_q^0) v_t^q(p_q^1)$$

Cette décomposition est motivée par la propriété majeure de la fonction exponentielle : le produit de deux exponentielles est l'exponentielle de la somme. Grâce à cette propriété il est facile de voir que la mise à jour s'écrit encore :

$$\forall (p_q^0, p_q^1) \in P_q^0 \times P_q^1, D_{t+1}(p_q^0, p_q^1) = a_{t+1}(q) v_{t+1}^q(p_q^0) v_{t+1}^q(p_q^1)$$

Avec

$$a_{t+1}^q = \frac{a_t^q Z_t^{0q} Z_t^{1q}}{Z_t}$$

$$v_{t+1}^q = \begin{cases} \frac{v_t^q(p_q) \exp(-\alpha_t h_t(p_q))}{Z_t^{1q}}, & \text{si } p_q \in P_q^1 \\ \frac{v_t^q(p_q) \exp(\alpha_t h_t(p_q))}{Z_t^{0q}}, & \text{si } p_q \in P_q^0 \end{cases}$$

L'algorithme suivant illustre les différentes étapes de l'adoption de RANKBOOST que nous avons proposée pour la recherche de passages pertinents dans les systèmes QR. Nous voyons que dans ce cas, la règle de mise à jour est linéaire au produit du

nombre de questions et du nombre d'exemples fois la complexité de l'algorithme de sélection.

3.5.3. *Expériences et Résultats*

Pour nos expériences, nous avons utilisé l'ensemble de questions et de réponses ainsi que la collection de documents AQUAINT utilisés lors de l'évaluation TREC-11. Comme nous utilisons une approche en deux étapes (appel du moteur de recherche à l'aide d'une requête puis ré-ordonnement des passages à l'aide de notre fonction apprise), le premier appel au moteur de recherche a été effectué en utilisant, pour chaque question, la requête obtenant la meilleure performance en terme de $a@n$ (answer-at- n : pourcentage des questions ayant au moins un passage contenant la réponse sur les n premiers passages renvoyés) sur la base d'apprentissage.

Les 250 premières questions TREC-11 nous ont servi de base d'apprentissage, et les 250 dernières ont été utilisées pour le test. Le moteur de recherche utilisé, ainsi que le découpage de la collection sont les mêmes que dans [CHA 02]. Le moteur de recherche utilisé est Managing Gigabyte (MG) [WIT 99]. Ces expériences ont pour but d'évaluer l'aptitude de la fonction apprise à ordonner les passages qui contiennent la réponse au-dessus de ceux qui ne la contiennent pas. Pour cette raison, nous avons enlevé des ensembles d'apprentissage et de test les questions qui n'ont pas de réponse dans les N premiers passages renvoyés par le moteur de recherche. Après filtrage, il reste 151 et 156 questions dans les bases d'apprentissage et de test respectivement.

Dans ces expériences, $N = 500$. Pour chacune des questions restantes, chaque passage renvoyé a été représenté comme un vecteur de dimension 117. L'algorithme proposé a été comparé au moteur de recherche MG. Les résultats sont présentés dans le tableau 1. La première colonne représente le seuil n , chacune des autres colonnes correspond alors au pourcentage des questions de l'ensemble de test pour lesquelles il y a au moins un passage contenant la réponse parmi les n premiers passages renvoyés. L'algorithme d'ordonnement obtient de très bonnes performances sur la base de test. La performance $a@n$ pour $n=5$ est comparable avec la performance $a@20$ du moteur de recherche. Lorsque $n=20$, le recouvrement est de l'ordre de 70%, et il atteint 90% lorsque $n=100$. Cela signifie par exemple que 109 des 156 questions de la base de test possèdent au moins un passage contenant la réponse lorsque $n=20$, qui est un seuil utilisé en pratique [GAI 03, MON 03].

 Algorithme RANKBOOST pour la recherche de passages pertinents

- Définir un ensemble de passages retournés par le moteur de recherche pour chaque question q et initialiser la distribution V_1^q ainsi que la distribution sur toutes les questions de la base d'apprentissage :

$$V_1^q(p_q) = \begin{cases} \frac{1}{|P_q^1|}, & p_q \in P_q^1 \\ \frac{1}{|P_q^0|}, & p_q \in P_q^0 \end{cases}$$

$$a_1^q = \frac{1}{|Q|}, \forall q \in Q$$

Pour $t=1, \dots, T$

- Apprendre une règle de base h_t suivant D_t .
- Mettre à jour D_t :

$$\forall (p_q^0, p_q^1) \in P_q^0 \times P_q^1, D_{t+1}(p_q^0, p_q^1) = a_{t+1}(q) v_{t+1}^q(p_q^0) v_{t+1}^q(p_q^1)$$

$$\forall q, a_{t+1}^q = \frac{a_t^q Z_t^{0q} Z_t^{1q}}{Z_t}$$

$$V_{t+1}^q = \begin{cases} \frac{v_t^q(p_q) \exp(-\alpha_t h_t(p_q))}{Z_t^{1q}}, & \text{si } p_q \in P_q^1 \\ \frac{v_t^q(p_q) \exp(\alpha_t h_t(p_q))}{Z_t^{0q}}, & \text{si } p_q \in P_q^0 \end{cases}$$

Où Z_t^{1q}, Z_t^{0q} et Z_t sont définis par :

$$Z_t^{0q} = \sum_{p_q \in P_q^0} v_t^q(p_q) \exp(\alpha_t h_t(p_q)), Z_t^{1q} = \sum_{p_q \in P_q^1} v_t^q(p_q) \exp(-\alpha_t h_t(p_q))$$

$$Z_t = \sum_{q \in Q} a_t^q Z_t^{0q} Z_t^{1q}$$

Fin Pour

$$\text{Sortie : } H = \sum_t \alpha_t h_t$$

Par rapport à l'état de l'art, [MON 03] a testé un moteur de recherche spécialement développé pour les QR, utilisant une fonction de score déterminée manuellement. Ses résultats sont les suivants: 52\% pour $n=5$, 70\% pour $n=20$ et 79\% pour $n=50$. Ces résultats sont à hauteur des nôtres. Cependant, il y a deux différences dans le protocole expérimental: pour l'évaluation de la fonction de score, il renvoie des documents entiers, et non des passages. Ce problème est plus facile que de renvoyer des passages. D'un autre côté, nous nous sommes limités aux questions de test ayant une réponse dans les 500 premiers documents renvoyés par MG, ce qui permet de penser que les questions que nous traitons sont les plus faciles.

D'une façon générale, d'un point de vue de l'application, nous ne pouvons conclure si la fonction apprise est supérieure ou non à l'état de l'art. Nous pouvons, par contre, conclure que l'algorithme a appris une combinaison performante, avec peu de temps de développement comparé à une méthode manuelle.

n premiers passages	MG(%)	Boosting (%) (IT=300)
1	10.2	22.4
5	24.3	46.1
10	32.7	58.9
20	46.1	69.9
60	51.3	75.6
40	56.4	84.6

Tableau 1 : Recouvrement des différents systèmes en % des 156 questions de la base test

3.6. Conclusion

Dans ce chapitre, nous avons présenté les principales méthodes d'apprentissage pour les systèmes QR. L'apprentissage permet des gains importants de généralité, par rapport aux corpus ou aux spécifications du système comme l'ensemble des classes de questions acceptées ou la langue dans laquelle les questions et les documents sont écrits. Il permet aussi, par le biais d'algorithmes génériques par rapport au nombre de caractéristiques utilisées, de combiner un grand nombre de critères hétérogènes pour déterminer un composant, ce qui peut amener une augmentation de performance du composant considéré.

Remerciements

Ce travail a été partiellement financé par le programme IST de la CE, dans le cadre du réseau PASCAL, IST-2002-506778. Cette publication reflète uniquement le point de vue des auteurs.

Bibliographie

[AGI 01] AGICHTEIN E., LAWRENCE S., GRAVANO L., *Learning search engine Query Transformations for Question Answering*, Proceedings of the 10th international conference on World Wide Web, 2001.

[AMI 00] AMINI M.R., ZARAGOZA H., GALLINARI P., *Learning for sequence Extraction Tasks*, Proceedings of the 6th Recherche d'Information Assistée par Ordinateur, (RIAO 2000), pp. 476-490, 2000.

[AND 04] ANDROUTSOPOULOS I., GALANIS D., *A Practically Unsupervised Learning Method to Identify Single-Snippet Answers to Definition Questions on the Web*, Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP 2005), Vancouver, Canada, 2005.

[BIK 97] BIKEL D.M., MILLER S., SCHWARTZ R., WEISCHEDEL R., *Nymble: High-Performance Learning Name-Finder*, Proceedings of the Fifth Conference on Applied Natural Language Processing, 1997.

[BUR 98] BURGES C.J.C., *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, volume 2(2), pages 121-167, 1998.

[CHA 02] DE CHALENDAR G., DALMAS T., ELKATEB-GARA F., FERRET O., GRAU B., HURAU-PLANTET M., ILLOUZ G., MONCEAUX L., ROBBA I., VILNAT A. *The Question Answering system QALC at LIMSI, experiments using the Web and WordNet*, Proceedings of Text REtrieval Conference, 2002.

[CHA 07] CHANG C.-C., LIN C.-J., *LIBSVM -- A Library for Support Vector Machines*, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

- [CHU 04] CHU-CAROLL J., CZUBA K., PRAGER J., BLAIR-GOLDENSOHN S., *IBM's PIQUANT II in TREC 2004*, Proceedings of Text REtrieval Conference, 2004.
- [COL 02] COLLINS M., DUFFY N., *Convolution Kernels for Natural Language*, Proceedings of Advances in Neural Information Processing Systems 14, 2002.
- [ECH 03] ECHIHABI A., HERMIAKOB U., HOVY E., MARCU D., MELZ E., RAVICHANDRAN D., *Multiple-Engine Question Answering in TextMap*, Proceedings of TREC 2003 conference, 2003.
- [EVE 01] EVEN-ZOHAR Y., ROTH D., *A sequential model for multiclass classification*, Proceedings of the SIGDAT Conference on Empirical Methods in Natural Language Processing (EMNLP-2001), pages 10-19, 2001.
- [FEL 98] FELLBAUM C.D., *WordNet, an Electronic Lexical Database*, MIT Press, Cambridge.
- [FRE 03] FREUND Y., IYER R., SCHAPIRE R.E., SINGER Y., *An Efficient Boosting Algorithm for Combining Preferences*, Journal of Machine Learning Research, volume 4, pages 933-969, 2003.
- [FRE 96] FREUND Y., SCHAPIRE R.E., *Experiments with a new Boosting Algorithm*, Proceedings of the 13th International Conference on Machine Learning, pages 148-156, 1996.
- [GAI 03] GAIZAUSKAS R., GREENWOOD M., HEPPLER M., ROBERTS I., SAGGION H., *Sargaison M., The University of Sheffield's TREC 2003 Q&A Experiments*, Proceedings of Text REtrieval Conference, 2003.
- [HAC 03] HACIOGLU K., WARD W., *Question classification with support vector machines and error correcting codes*, Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003--short papers - Volume 2, pages 28-30, 2003.
- [HAR 01] HARABAGIU S., BUNESCU R., MAIORANO S., *Text and knowledge mining for coreference resolution*, Proceedings of the Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001, pages 1-8, 2001.
- [HAR 03] HARABAGIU S., MAIORANO S., PASCA M.A., *Open-domain Textual Question Answering Techniques*, Natural Language Engineering, volume 1(1), pages 1-38, 2003.
- [HER 97] HERMIAKOB U., MOONEY, R.J., *Learning parse and translation decisions from examples with rich context*, Proceedings of the 35th annual meeting on Association for Computational Linguistics, 1997.
- [HER 01] HERMIAKOB U., *Parsing and question classification for question answering*, Proceedings of the workshop on ARABIC language processing: status and prospects - Volume 12, pages 1-6, 2001.
- [ITT 00] ITTYCHERIAH A., FRANZ M., ZHU W.J., RATNAPARKHI A., MAMMONE R.J., *IBM's Statistical Question Answering system*, Proceedings of Text REtrieval Conference, 2000.
- [ITT 01] ITTYCHERIAH A., FRANZ M., ROUKOS S., *IBM's Statistical Question Answering System - TREC-10*, Proceedings of Text Retrieval Conference (TREC), 2001.

- [JIJ 04] JIJKOUN V., DE RIJKE M., *Answer selection in a multistream open domain question answering system*, Proceedings of European Conference on Information Retrieval, 2004.
- [KOL 06] KO J., HIYAKUMOTO L., NYBERG E., *Exploiting Multiple Semantic Resources for Answer Selection*, Proceedings of International Conference on Language Resources and Evaluation, 2006.
- [LIH 05] LI X., HUANG X.-J., WU L.-D., *Question Classification using Multiple Classifiers*, Proceedings of the Second International Joint Conference on Natural Language Processing: Companion Volume including Posters/Demos and tutorial abstracts, 2005.
- [LIR 02] LI X., ROTH D., *Learning question classifiers*, Proceedings of the 19th International Conference on Computational Linguistics, 2002.
- [MET 05] METZLER D., CROFT W.B., *Analysis of Statistical Question Classification for Fact-Based Questions*, Information Retrieval, volume 8(3), pages 481-504, 2005.
- [MIL 04] MILIARAKI S., ANDROUTSOPOULOS I., *Learning to Identify Single-Snippet Answers to Definition Questions*. Proceedings of Conference on Computational Linguistics, 2004.
- [MON 03] MONZ C., *Document Retrieval in the Context of Question Answering*, Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03), 2003.
- [NGK 01] NG W.T., KWAN J., XIA Y., *Question Answering Using a Large Text Database: A Machine Learning Approach*, Proceedings of the SIGDAT Conference on Empirical Methods in Natural Language Processing (EMNLP-2001), 2001.
- [PEN 05] Peng F., Weischedel R., Lieuana A., Xu J., *Combining deep linguistics analysis and surface pattern learning: a hybrid approach to Chinese definitional question answering*, Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP), 2005.
- [PER 04] PÉREZ-COUTIÑO M.A., SOLORIO T., MONTES-Y-GÓMEZ M., LÓPEZ-LÓPEZ A., PINEDA L.V., *Question Answering for Spanish Based on Lexical and Context Annotation*, IBERAMIA, pages 325-333, 2004.
- [PUN 01] PUNYAKANOK V., ROTH D., *The use of classifiers in sequential inference*, Proceedings of Advances in Neural Information Processing Systems 13, 2001.
- [PUR 07] PURWARIANTI A., TSUCHIYA M., NAKAGAWA S., *A Machine Learning Approach for Indonesian Question Answering System*, Proceedings of Artificial Intelligence and Applications, 2007.
- [RAM 04] RAMAKRISHNAN G., CHAKRABARTI S., PARANJPE D., BHATTACHARYA P., *Is question answering an acquired skill ?*, Proceedings of the 13th international conference on World Wide Web (WWW'04), 2004.
- [RAV 02] RAVICHANDRAN D., HOVY, E., *Learning Surface Text Patterns for a Question Answering System*, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL), 2002.
- [RAV 03] RAVICHANDRAN D., HOVY E., OCH F.J., *Statistical QA - Classifier vs Re-ranker: What's the difference ?*, Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering--Machine Learning and Beyond, 2003.

- [ROT 01] ROTH D., KAO G.K., LI X., NAGARAJAN R., PUNYAKANOK V., RIZZOLO N., YIH W.-T., ALM C. O., MORAN L.G., *Learning components of a Question-Answering System*, Proceedings of Text REtrieval Conference, 2001.
- [SCH 90] SCHAPIRE R.E., *The Strength of Weak Learnability*, Machine Learning Journal, Volume 5, Numéro 2, pp. 197-227, 1990.
- [SOL 04] SOLORIO T., PÉREZ-COUTIÑO M., MONTES-Y-GÉMEZ M., VILLASEÑOR-PINEDA L., LÓPEZ-LÓPEZ A., *A language independent method for question classification*, Proceedings of the 20th international conference on Computational Linguistics, 2004.
- [SOU 02] SOUBBOTIN M.M., SOUBBOTIN S.M., *Use of patterns for detection of answer strings: a systematic approach*, Proceedings of Text Retrieval Conference (TREC), 2002.
- [SUZ 02] SUZUKI J., SASAKI Y., MAEDA E., *SVM answer selection for open-domain question answering*, Proceedings of the 19th international conference on Computational linguistics, pages 1-7, 2002.
- [TEL 03] TELLEX S., KATZ B., LIN J., MARTON G., FERNANDES A., *Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering*, Proceedings the 26th international ACM SIGIR conference on Research and development in information retrieval, 2003.
- [USU 06] USUNIER N. *Apprentissage de fonctions d'ordonnement: une étude théorique de la réduction à la classification et deux applications à la Recherche d'Information*. Thèse de Doctorat, Université Pierre et Marie Curie, Paris 6, 2006.
- [USU 04] USUNIER N., AMINI M.R., GALLINARI P., *Boosting Weak Ranking Functions to Enhance Passage Retrieval for Question Answering*, Proceedings the SIGIR 2004 workshop on Information Retrieval for Question Answering, 2004.
- [VAP 00] VAPNIK V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 2000.
- [WIT 99] WITTEN I.H., MOFFAT A., BELL T.C., *Managing Gigabyte: Compressing and Indexing Documents and Images*, Morgan Kaufmann, San Francisco, 1999.
- [ZHA 03] ZHANG D., LEE W.S., *Question classification using support vector machines*, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, pages 26-32, 2003.