Combinaison de fonctions de préférence par Boosting pour la recherche de passages dans les systèmes de question/réponse

Nicolas Usunier, Massih-Reza Amini, Patrick Gallinari

Laboratoire d'Informatique de Paris 6
8, rue du Capitaine Scott, 75015 Paris
{usunier, amini, gallinari}@poleia.lip6.fr

Résumé. Nous proposons une méthode d'apprentissage automatique pour la sélection de passages susceptibles de contenir la réponse à une question dans les systèmes de Question-Réponse (QR). Les systèmes de RI *ad hoc* ne sont pas adaptés à cette tâche car les passages recherchés ne doivent pas uniquement traiter du même sujet que la question mais en plus contenir sa réponse. Pour traiter ce problème les systèmes actuels ré-ordonnent les passages renvoyés par un moteur de recherche en considérant des critères sous forme d'une somme pondérée de fonctions de scores. Nous proposons d'apprendre automatiquement les poids de cette combinaison , grâce à un algorithme de réordonnencement défini dans le cadre du *Boosting*, qui sont habituellement déterminés manuelement. En plus du cadre d'apprentissage proposé, l'originalité de notre approche réside dans la définition des fonctions allouant des scores de pertinence aux passages. Nous validons notre travail sur la base de questions et de réponses de l'évaluation TREC-11 des systèmes de QR. Les résultats obtenus montrent une amélioration significative des performances en terme de rappel et de précision par rapport à un moteur de recherche standard et à une méthode d'apprentissage issue du cadre de la classification.

1 Introduction

Les systèmes de question/réponse (QR) ont pour objectif de trouver la réponse à une question formulée en langage naturel dans un grand corpus de documents. Nous nous intéressons ici aux systèmes de QR en domaine ouverts, développés dans le cadre des évaluations TREC[1]. Dans ces systèmes, le traitement d'une question s'effectue en trois étapes : **(1)** l'analyse la question, déterminant un type de réponse attendue et la structure syntaxique de la question, **(2)** la recherche d'information (RI), qui interroge un moteur de recherche pour sélectionner des passages susceptibles de contenir la réponse à la question. Selon les systèmes, les passages peuvent être des documents entiers (Monz 2003), des parties de documents de longueur fixe (Chalendar et al. 2002), ou des phrases consécutives d'un document (Prager et al. 2000). Enfin, **(3)** l'extraction et la sélection de la réponse dans les passages séléctionnés.

Dans la chaîne de traitement, le module de RI est crucial, car s'il échoue à renvoyer au moins un passage contenant la réponse dans sa sélection, le système ne peut pas répondre à la question. Par ailleurs, il pose de nouveaux problèmes de RI : la recherche qu'il doit effectuer est plus spécifique

[1] Text REtrieval Conference, http ://trec.nist.gov

que dans la RI *ad-hoc* qui recherche des passages traitant du même sujet que la requête utilisateur (dans notre cas la question), mais qui ne contiennent pas nécessairement la réponse à la question. De plus, bien que la probabilité qu'au moins un des passages renvoyés conntienne la réponse (appelée *recouvrement* dans la suite) augmente avec le nombre de passages sélectionnés, (Gaizauskas et al. 2003) montrent que si l'on sélectionne trop de passages, cela peut nuire aux performances globales du système car cela introduit du bruit dans le processus de sélection de la réponse. Il y a donc un compromis à trouver, qui est d'avoir un fort recouvrement lorsque peu de passages (quelques dizaines) sont sélectionnés (Monz 2003, Gaizauskas et al. 2003).

Pour résoudre ce problème de RI, de nombreux auteurs proposent une approche en deux étapes. Tout d'abord, un appel à un moteur de recherche sélectionne un grand nombre N de passages, afin d'avoir un fort recouvrement. Ensuite, ces passages sont ré-ordonnés en prenant en compte des critères favorisant les passages contenant la réponse. Les n ($n << N$) passages les mieux classés après ré-ordonnancement sont envoyés à l'étape d'extraction de réponse. Une étude comparative des différentes approches suivant ce schéma est disponible dans (Tellex et al. 2003). Les critères de ré-ordonnancement peuvent être la similarité à une requête, la présence d'expressions ou de mots-clefs caractéristiques de la question dans le passage, ainsi que la distribution des mots dans le passage. Ils sont actuellement pris en compte par le biais de *fonctions de préférence* qui allouent un score par rapport à un critère particulier. La fonction globale est une somme pondérée des fonctions de préférence, dont les poids sont déterminés à la main. Le nombre de fonctions à combiner doit donc être faible (autour de cinq), et la modification d'une fonction de préférence nécessite de déterminer de nouveaux poids, donc un travail long et fastidieux.

L'apprentissage automatique des poids de la combinaison linéaire rendrait possible d'apprendre des combinaisons d'un plus grand nombre de fonctions de pertinence, donc de prendre en compte plus de critères. D'autre part, il permettrait d'acquérir de la généricité et de l'évolutivité au niveau de la recherche de passages dans les systèmes de QR : il suffirait de ré-entraîner le modèle pour inclure dans la combinaison de nouvelles fonctions, en enlever ou les modifier. Ainsi, une telle approche pourrait amener à un développement plus rapide de cette étape des systèmes de QR, en augmentant sa généricité et son évolutivité tout en optimisant ses performances.

Dans cet article, nous proposons une méthode pour l'apprentissage automatique des poids de la combinaison. Nous montrons qu'une approche simple de classification est insuffisante, et proposons l'utilisation d'un autre type d'algorithmes d'apprentissage, dits d'ordonnancement, qui apprennent une combinaison des fonctions de préférence à partir d'une relation d'ordre partiel sur les exemples, et non une information de classe. Nous avons utilisé l'algorithme *RankBoost* (Freund et al. 2003) qui a été utilisé avec succès dans des problèmes de *métarecherche* et de *filtrage collaboratif*. Nous pensons que le cadre offert par ce type d'algorithmes est aussi adapté à la tâche de recherche de passage dans les systèmes de QR. De plus, pour montrer comment un grand nombre de critères peuvent être pris en compte sans avoir à fournir un travail humain important, nous proposons un algorithme original de génération automatique des fonctions de préférence qui permet d'en obtenir un grand nombre, donc de considérer un grand nombre de critères dans la combinaison.

Cet article est organisé comme suit. Tout d'abord, nous décrivons la génération des fonctions de préférence dans la section 2. Nous détaillons ensuite le modèle proposé (section 3). Nous présentons enfin l'évaluation de notre approche sur le corpus de questions et de réponses de l'évaluation `TREC-11` dans la section 4.

2 Fonctions de préférence

Nous allons dans cette section présenter notre méthode de génération automatique des fonctions de préférence qui assignent des scores aux passages. Chaque passage est ensuite décrit par un vecteur de taille fixe où chacune de ses composantes est un score donné par chacune de ces fonctions. Le but est alors d'entraîner un modèle qui va prendre en entrée un vecteur (constitué de ces scores de bases) décrivant un passage et qui en sortie donne un score global qui est une combinaison des diffèrents scores de bases. Les passages sont alors triés dans l'ordre décroissant de leur score global.

Notre générateur de fonctions de préférence est fondé sur des *règles de formulation de requêtes* (RFR). Une RFR est une règle qui génère une requête étant donné une question. Dans notre cas, nous les représenterons comme des listes de doublets *[(règle d'extension de mot, catégorie de mots)]*. Pour une question donnée, une RFR renvoie une requête composée des mots obtenus en appliquant la *règle d'extension de mot* à tout les mots de la *catégorie de mots* correspondante. Pour créer ces doublets nous utilisons (1) les liens de l'ontologie WordNet (Fellbaum 1998) pour créer les *règles d'extension* (2) l'analyseur de questions utilisé dans le système QALC (Chalendar et al. 2002) pour créer des *catégories de mots*. Ces catégories correspondent à des critères syntaxiques ou morpho-syntaxiques. Par exemple, 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 (Usunier et al. 2004). A partir de ces RFR, nous avons obtenu 112 fonctid'excellence ons de préférence grâce à l'association suivante : A chaque RFR $\mathcal{R} : \{questions\} \longmapsto \{requetes\}$ est associé une fonction de préférence $f_{\mathcal{R}} : \{questions\} \times \{passages\} \longmapsto \mathbb{R} \cup \{\perp\}$ telle que $f_{\mathcal{R}}(q, p) = \text{score}(\mathcal{R}(q), p)$ si p est dans les N premiers passages renvoyés et $\perp$ sinon. Ici, $\perp$ signifie que $f_{\mathcal{R}}$ n'alloue aucun score au passage p. La fonction $score$ que nous avons utilisé dans nos expérience est la mesure cosinus du moteur de recherche MG (Witten et al. 1999).

A cet ensemble de 112 caractéristiques, nous avons ajouté 5 autres fonctions de préférence inspirées du système QR d'IBM (Ittycheriah et al. 2000) et évaluées comme étant parmi les plus performantes dans l'étude comparative de (Tellex et al. 2003).

3 Boosting pour l'ordonnancement de passages

Pour chaque question, une liste de passages a été créée en interrogeant le moteur de recherche avec la requête générée par la RFR qui a obtenu les meilleures performances en terme de $a@n$[2] parmi celles que l'on a sélectionnées. Notre objectif est alors d'ordonner les passages qui contiennent la réponse pour une question donnée au-dessus de ceux qui ne la contiennent pas. Pour cela nous avons utilisé l'algorithme *RankBoost* (Freund et al. 2003). Cet algorithme a été spécifié pour apprendre à ordonner les éléments en combinant des ensembles de fonctions de préférence. La sortie de l'algorithme est une fonction assignant des scores aux exemples, qui permet ainsi de les ordonner. L'information utilisée par l'algorithme pour apprendre une telle fonction est un ordre partiel sur l'ensemble d'apprentissage. Etant donné l'ensemble des paires de passages pour lesquelles cet ordre partiel est défini, le critère que minimise *RankBoost* est le nombre de paires de passages pour lesquelles l'ordre induit par la fonction de score apprise est incorrect. Le principe de *RankBoost* est

[2] *answer-at-n*, qui est le recouvrement lorsqu'on garde les n passages les mieux class´es apr`es le re-ordonnancement.

de pondérer itérativement les exemples et d'entraîner un algorithme d'apprentissage simple (*weak learner*) à partir de cette distribution. Le fait de combiner des règles de décisions dérivées des fonctions de préférence est la technique utilisée dans (Freund et al. 2003). Nous l'avons utilisée aussi car elle présente une faible complexité, bien que d'autres fonctions plus complexes peuvent être implémentées pour être combinées dans la fonction finale.

4 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` . Les 250 premières questions 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 (Chamendar et al. 2002). Le moteur de recherche utilisé est *Managing Gigabyte (MG)* (Witten et al. 1999). 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 à deux autres systèmes : le moteur de recherche MG qui donne une liste ordonnée de passages, ainsi qu'une machine à vecteurs supports (SVM) linéaire. Ce dernier système est largement utilisé dans des tâches de classification bi-classes (Joachims 1998). Dans notre cas, le SVM a été entraîné pour classer les passages comme contenant la réponse ou non, correspondant aux classes 1 et 0 respectivement. L'entrée du SVM est la même que l'entrée de *RankBoost*, sauf que nous avons fixé la valeur de caractéristique 0 lorsqu'une fonction de pertinence n'associait aucun score à un passage. Pour notre tâche d'ordonnancement nous avons utilisé comme score de pertinence la distance d'un passage à l'hyperplan séparateur trouvé par le SVM. Nous comparons dans le tableau 1 la mesure $a@n$ spécifique pour les systèmes de QR, pour les trois systèmes, en fonction de n : le moteur de recherche MG, le SVM linéraire et l'algorithme de boosting. 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. Le SVM obtient de faibles performances

n premiers passages	MG $(\%)$	SVM $(\%)$	Boosting $(\%)$ (IT=300)
1	10.2	5.1	**22.4**
5	24.3	19.2	**46.1**
10	32.7	33.9	**58.9**
20	46.1	46.1	**69.9**
30	51.3	50.6	**75.6**
40	56.4	55.7	**84.6**

TAB. 1 – Recouvrement des différents système en % des 156 questions de la base de tests.

en généralisation. Ces performances laissent supposer que les deux ensembles de passages **(1)** qui

contiennent la réponse et **(2)** qui ne la contiennent pas, ne sont pas séparables dans l'espace des caractéristiques considéré.

Cela est probablement dû au fait que pour chaque question, les ensembles de passages renvoyés sont indépendants mais qu'ils se situent dans l'espace des caractéristiques dans des régions différentes selon les questions. Ainsi, chercher une séparation globale des exemples, indépendante des questions, est une tâche difficile. Par contre, l'algorithme de boosting 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%. Cela signifie par exemple que 109 des 156 questions de la base de test possèdent au moins un passage contenant la réponse lorque $n = 20$, qui est un seuil utilisé en pratique (Gaizauskas et al. 2003, Monz 2003). Bien sûr, l'algorithme présenté est générique, et des raffinements supplémentaires sur les caractéristiques permettraient probablement d'accroître encore les performances. Comme l'algorithme de boosting cherche à discriminer les passages deux à deux, il ne recherche pas de frontière de décision globale. Les résultats indiquent la différence en terme d'apprentissage entre les tâche de classification et d'ordonnancement, et le fait que les algorithmes de classifications sont mal adaptés à la tâche d'ordonnancement. Une raison à cela peut être qu'un algorithme de classification recherche une séparation entre les passages contenant la réponse et ceux qui ne la contiennent pas, cette séparation étant indépendante de la question. Or, dans les faits, il semble qu'une telle séparation n'existe pas. Au lieu de cela, un algorithme d'ordonnancement, qui recherche des séparations locales entre les exemples, semble beaucoup plus adapté à la tâche de recherche de passages dans les systèmes de QR.

5 Conclusion

Dans cet article nous avons proposé l'utilisation d'algorithmes d'ordonnancement pour la tâche de recherche de passages dans les systèmes de QR, et montré leur efficacité par rapport à d'autres méthodes. D'autre part nous avons proposé une méthode de génération automatique de fonctions de préférence pour les systèmes de QR. Des fonctions de préférence plus sophisitiquées ainsi que d'autres algorithmes d'ordonnancement sont encore à étudier, qui laissent espérer une amélioration des résultats. Le modèle doit aussi être étudié et testé sur d'autres collections de questions et de documents, et sur des corpus de questions hétérogènes.

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.

Références

de Chalendar G., Dalmas T., Elkateb-Gara F., Ferret O., Grau, Hurault-Plantet M., Illouz G., Monceaux L., Robba I., Vilnat A. (2002), The Question Answering system QALC at LIMSI, experiments using the Web and WordNet, Proceedings of TREC 2002.

Fellbaum C.D. (1998), WordNet, an Electriconic Lexical Database, MIT Press, Cambridge.

Freund Y., Iyer R., Schapire R.E., Singer Y. (2003) An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research, 4 :933-969.

Gaizauskas R., Greenwood M.A. Hepple M. Roberts I., Saggion H., Sargaison M. (2003), The University of Sheffield's TREC 2003 Q& A Experiments, Proceedings of TREC 2003.

Harabagiu S., Moldovan D., Pasca M., Mihalcea R., Surdeanu M., Bunescu R., Girju R., Rus V., Morarescu P. (2001), The role of lexico-semantic feedback in open-domain textual question-answering, Proceedings of ACL-2001.

Ittycheriah A., Franz M., Zhu W.-J., Ratnaparkhi A., Mammone R.J. (2000), IBM's Statistical question answering system, Proceedings of TREC 2000.

Joachims T. (1998) Text Categorization with support vector machines : learning with many relevant features, Proceedings of the 10th European Conference on Machine Learning.

Katz B., Lin J., Loreto D. Hildebrandt W., Bilotti M., Felshin S., Fernandes A., Marton G., Mora F. (2003), Integrating Web-based and corpus-based techniques for question answering, Proceedings of the 12th Text REtrieval Conference (TREC 2003).

Moldovan D., Harabagiu D., Girju R., Morarescu P., Lacatusu F., Badulescu A., Bolohan O. (2002), LCC Tools for Question Answering, Proceedings of TREC 2002

Monz C. Document Retrieval in the context of question answering (2003), Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03).

Tellex S., Katz B., Lin J., Marton G. Fernandes A. (2003) Quantitative evaluation of passage retrieval algorithms for question answering, Proceedings SIGIR 2003.

Usunier N., Amini M., Gallinari P. (2004) Boosting Weak Ranking Functions to Enhance Passage Retrieval for Question Answering, In IR4QA workshop of SIGIR 2004

Witten I.H., Moffat A.,Bell T.C. (1999) Managing Gigabyte : Compressing and Indexing Documents and Images, Morgan Kaufmann, San Fransisco.

Summary

We investigate the problem of passage retrieval for Question Answering (QA) systems. We adopt a machine learning approach and apply to QA a boosting algorithm initially proposed for ranking a set of objects by combining baseline ranking functions. The system operates in two steps. For a given question, it first retrieves passages using a conventional search engine and assigns each passage a series of scores. It then ranks the returned passages using a weighted feature combination. Weights express the feature importance for ranking and are learned to maximize the number of top ranked relevant passages over a training set. We empirically show that using questions from the `TREC-11` question/answering track and the `Aquaint` collection, the proposed algorithm significantly increases both coverage and precision with respect to a conventional IR system.