Recherche des Services Web en Utilisant le Contenu d'OWLS

FETHALLAH Hadjila[1], CHIKH Mohammed Amine[2]
Université de Abou Bekr Belkaid Tlemcen
[1]f_hadjila mail.univ-tlemcen.dz
[2]mea_chikh @mail.univ-tlemcen.dz

Résumé. L'objectif de cet article est de résoudre le problème de matching (recherche) des services web. Dans ce cadre nous présentons une approche basée sur le contenu des documents OWLS. Tout d'abord nous supposons que l'ensemble des services web de la collection de test est segmenté en sept classes. Apres nous représentons chaque service, par un vecteur de fréquences des termes, ceci peut être fait en exploitant les descriptions informelles (en langage naturel) contenues dans les documents OWLS, après nous exploitons, ces vecteurs de fréquences et les concepts d'entrées-sorties associés aux couples (requête, service web), ainsi que de la mesure de similarité cosine, pour faire l'appariement (matching). Les résultats obtenus ont été largement acceptables, et encouragent les travaux de recherche sur cette piste.

Mots clés : Recherche de services web, Mesures de similarité, ontologies, Owls, text mining.

Introduction

Les services web se présentent aujourd'hui comme un support crédible permettant à des applications d'exposer leurs fonctionnalités au travers d'interfaces standardisées et de plus en plus éprouvées. Cette technologie permet de créer des systèmes d'informations interopérables (échangeant des données) et faiblement couplés (peu sensibles aux changements), son principe se fonde sur la possibilité d'invoquer une application sur la toile mondiale, en utilisant les protocoles d'Internet, et en faisant abstraction aux caractéristiques d'implémentation du service (la plateforme, le langage de programmation, les API).

L'épine dorsale des services Web est articulée sur les trois standards suivants: WSDL, UDDI et SOAP [9], [8]. Ces protocoles facilitent la description, la publication, la découverte et la communication entre services. Cependant, cette infrastructure de base ne permet pas encore aux services Web d'assurer une

interopérabilité automatisée. Cette automatisation est pourtant essentielle pour faire face aux exigences de passage à l'échelle, et pour réduire les coûts de développement.

Le premier pas de cette interopérabilité commence au niveau de la recherche automatique des services web. En effet, lorsqu'on peut localiser efficacement les services web dont on a besoin, on peut réaliser par la suite, les autres formes de l'interopérabilité (telles que la composition, l'invocation, la surveillance…)

Différentes approches ont été développées pour résoudre la problématique de recherche [3],[11],[15],[20]. Dans le présent article, nous considérons trois parties du document OWLS [10], dans la description d'un service web : le « profile textual description », le profile:hasInput, et le profile:hasOutput. Le premier élément est un fragment qui contient des informations textuelles d'ordre fonctionnel (entrées-sorties) et non fonctionnel (qualité de service). Cette description informelle est ensuite prétraitée et segmentée en classes (selon la thématique du service web).

L'opération de recherche d'un service web est établie en deux phases, nous sélectionnons d'abord la ou les classes de services web, les plus proches à notre requête, ensuite nous comparons chacun de leurs membres avec le besoin spécifié par L'utilisateur (ie le vecteur des fréquences des termes de la requête).

Notre contribution peut être résumée comme suit :

- préparation des textes (les fragments « profile textual description ») à classifier: suppression des mots vides, stemming.

- création des vecteurs (les sacs de mots) qui modélisent les fréquences (nombre d'occurrences) des termes de chaque texte prétraité.

- calcul de la moyenne des vecteurs de fréquences de chaque classe de services web.(dans ce cas là, nous calculons 07 vecteurs moyens).

- prétraitement et modélisation de la requête de l'utilisateur en la considérant comme un texte brut (suppression des mots vides, stemming, représentation en terme de vecteur de fréquences de termes).

- comparaison du vecteur de la requête avec les différents vecteurs de fréquences qui modélisent les classes, nous retenons ceux qui dépassent un certain seuil. La mesure utilisée est celle de cosine.

- comparaison de chaque membre des catégories retenues précédemment avec la requête, en utilisant la mesure de similarité « cosine ». Pour cela nous utilisons trois

descripteurs : le « profile textual description », le profile:hasInput, et le profile:hasOutput.

Nous avons choisi l'utilisation de la mesure de similarité cosine parce qu'elle est considérée comme l'une des mesures les plus proéminentes dans la recherche d'information [5].

Le reste de l'article est structuré comme suit : la deuxième section montre un état de l'art sur la recherche des services web. La troisième section introduit le standard OWLS et les principaux concepts sous jacents. La quatrième section présente l'approche développée, la cinquième section montre les résultats obtenus, et nous terminons enfin par une conclusion et quelques perspectives.

1. Etat de l'art

Plusieurs MachMakers ont été développé ces dernières années, Nous citons OWLSUDDI matchmaker [19], RACER [16], SDS [17], MAMA [6], HotBlu [7], [14], et OWLS-MX, [15], la majorité d'entre eux utilise les éléments input/outputs du profile dans leur processus de matching.

D'autres approches [4] utilisent le process-model matching, les auteurs de la référence [2] adoptent le matching d'arbres récursifs. On développe dans [1] une méthode de découverte basée sur un réseau P2P, le projet WSMO (web service modeling ontology) [13] adopte un ensemble de médiateurs qui assurent la recherche et la composition des services web, de même on propose dans [20] une approche nommée METEOR-S qui se base sur l'extension sémantique de WSDL pour réaliser la recherche.

Nous notons que les prototypes LARKS [18] et OWLS-MX sont deux systèmes de recherche hybride, parce qu'ils combinent les capacités formelles (basées sur les logiques de descriptions) et informelles des services web, dans le matching. OWLS-MX propose 05 types de processus de matching, dont un est purement logique, et les autres sont des méthodes inspirées des techniques de recherche d'information.

2. OWLS (ontology web language services)

OWLS [10] est une ontologie de haut niveau qui indique qu'une ressource (entreprise) est liée à un service, à son tour le service est constitué d'un profile, d'un service model et d'un service grounding. En bref, le profile indique ce que fait le service, le service model décrit le comment : les étapes qui composent le service et le flot de control, et le service grounding décrit la manière d'accéder au service. Ces trois concepts sont conçus pour donner une vue globale sur les capacités du

service. Nous notons que le service profile contient un élément donnant une description textuelle (informelle) du service web

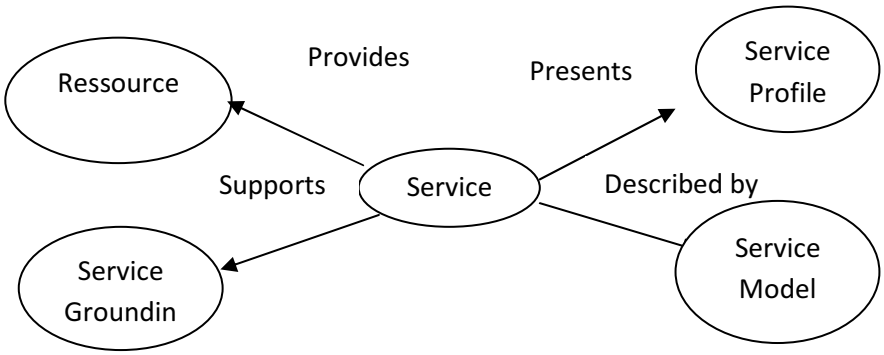

Figure 1 : ontologie de haut niveau d'OWLS

2.1. Service profile

Le « profile» [10] décrit ce que fait le service. Un système recherchant un service examinerait la première fois le « profile» pour voir si le service fournit ce qui est nécessaire. Le « profile» contient les informations suivantes:

• Le nom du service, les contacts et la description textuelle qui sont communément appelés propriétés non fonctionnelles.

• La description de fonctionnalité "IOPE" (inputs, outputs, preconditions, effects).

• Une classification selon une taxonomie industrielle et une description de qualité. Le nom de service peut être employé comme identification, alors que les descriptions de contact et de service sont du texte destiné spécialement pour des humains. La description de fonctionnalité permet au « profile» d'exposer les entrées, les sorties, les effets et les conditions préalables du service. En fin, le « profile» contient aussi de l'information sur la classification de service dans les diverses taxonomies, et quelques attributs pour décrire la qualité des services.

2.2. Service Model

Le modèle de service [10], décrit le fonctionnement interne du service en terme de processus. Cette partie décrit la transformation entreprise par le service à travers Les données (les entrées et les sorties) et la transformation d'état (les préconditions et les effets.) Le modèle de service prévoit des primitives pour

exprimer les différents types de processus, et des relations de contrôle. Plus précisément owls prévoie les types suivants : Processus Atomique, Processus Composé, et Processus Simple.

Le processus atomique est exécutable dans une seule étape. Il représente le plus petit module qui sert à la création des autres processus, et il ne contient pas de sous processus en interne.

Le processus composé : signifie que le processus peut être composé de sous-processus. Un ensemble de constructeurs est utilisé pour spécifier le flot de control et le flot de données tels que la séquence, les primitives de choix, de split …

Un processus simple n'est pas exécutable (ou invocable). Il fournit une vue abstraite d'un processus ou d'un a ensemble de processus composés.

2.3. Service Grounding

Le grounding décrit l'accès au service [10]. Il permet de spécifier les protocoles de transport et les formats de message. Le « profile» et le modèle de service sont considérés comme des représentations abstraites du service. Le rôle du grounding est de transformer ces représentations abstraites en une forme concrète qui peut être employée pour l'interaction. OWLS repose sur WSDL pour spécifier l'interaction des services. Ainsi WSDL peut être considéré comme une couche basse utilisée dans owls.

3. Approche Proposée

Les données utilisées dans nos expérimentations sont issues d'un échantillon du corpus owls-tc version 2.2.1 Cette base est développée par le centre allemand pour la recherche en intelligence artificielle (http://www.dfki.de/scallops). Cette base décrit un ensemble de services web à travers des documents owls. Ces documents comportent une partie exprimée en langage naturel. Elle jouera le point de départ de notre processus de création de vecteurs de fréquences de termes (indexation). L'ensemble des documents est reparti en sept classes : l'économie, la communication, l'éducation, l'alimentation, le domaine militaire, le domaine médical.

Nous notons que la requête est aussi modélisée comme un service web, cad qu'elle possède un document OWLS avec un « profile textual description » , un « profile :hasinput » et un « profile :hasoutput ». Tous ces descripteurs sont utilisés lors de la recherche.

L'approche proposée est composée de deux parties : une phase d'indexation basée sur les techniques du textmining, et une phase de recherche basée sur la mesure cosine

Le processus d'indexation de l' élément « profile textual description » est organisé comme suit :

- La suppression des signes de ponctuation et des mots vides tels que les articles et les prépositions.

- La détection des stems des mots (élimination des suffixes et des préfixes) pour minimiser le vecteur représentatif.

- Le calcul du nombre d'occurrence de chaque terme et remplissage du vecteur (la taille du vecteur est égale au nombre total des différents stems possibles).

- le calcul la moyenne des vecteurs de fréquences de chaque classe de services web.

La base de recherche contient 56 exemples étiquetés en 7 classes, chaque classe possède 8 instances. La taille de chaque vecteur est égale à 226 attributs (qui forment les différents stems de la base). Chaque classe possède une ontologie de domaine.

Le processus de recherche est réalisé comme suit :

- tout d'abord la requête est modélisée sous forme d'un sac de mots (vecteur de fréquences de termes), en utilisant son propre « profile textual description ».

- nous sélectionnons la ou les classes les plus proches par rapport à notre requête en utilisant la mesure cosine. si on considère que A est une requête et B est un vecteur moyen de fréquences de termes (ie 02 sacs de mots) alors : cos (A,B)= <A,B>/(||A|| . || B||). avec <A,B> : le produit scalaire de A et B. nous retenons les classes (vecteurs moyens) dépassant un seuil de matching. (le seuil est déterminé expérimentalement).

- nous comparons ensuite la requête avec chaque membre des classes retenues précédemment, en utilisant l'algorithme suivant :

    - soit InputR ={Ir1,Ir2,…Irn} et OutputR ={Or1,0r2,…,Irm} les inputs et les outputs de la requête R.

- De même InputS ={Is1,Is2,…Isp} et OutputS ={Or1,0r2,…,Isq} les inputs et les output du service S.

- Nous augmentons chaque élément des quatre ensembles précédents avec tous ses ascendants (subsumants) dans l'ontologie de domaine. Par exemple si Ir1 possède c1 et c2 comme concepts ascendants dans l'ontologie alors InputR = {Ir1,Ir2,…Irn, c1,c2}

- Pour toutes les classes retenues i faire:

  - resultat1= {S tel que S∈ classe i et (cos(InputR, InputS) + cos(OutputR, OutputS))/2 ≥ θ1 }

  - resultat2= {S tel que S∈ classe i et cos(VS, Req) ≥ θ2}

  - les paramètres θ1, θ2 sont choisis pendant l'expérimentation. Et VS dénote le vecteur des fréquences de termes associé à S. et Req={f'1,..f'k} le vecteur de fréquences de termes de la requête.

  - résultat = resultat1∪ resultat2.

  - Fin_faire

- nous évaluons les résultats à travers les critères de rappel et de précision.

4. Expérimentation

4.1. Introduction

Nous avons réalisé, le prototype de l'approche sous la plateforme windows xp, sur machine munie d'un processeur Intel Pentium M 1,75 GHZ avec 512 Mo de RAM.

Nous allons définir par la suite les critères de rappel et de précision utilisés dans la l'évaluation des résultats.

La précision P d'une classe i est la faculté de diminuer le nombre de résultats affectés par erreur à i. Et plus formellement :

P = vp/(vp+fp). Avec :

vp : les vrais positifs. (Un résultat correct, et considéré comme juste par le système).

fp : les faux positifs. (Un résultat erroné, mais considéré comme juste par le système).

Le rappel R associé à une classe i, est la faculté de ne pas affecter un exemple d'une classe i à une fausse classe j, et plus formellement :

R=vp/(vp+fn). avec :

vp : les vrais positifs. (Un résultat correct, et considéré comme juste par le système).

fn : les faux négatifs. (Un résultat correct, et considéré comme faux par le système).
La F-mesure est combinaison du rappel et de la précision :

F-mesure= (2*rappel*précision) /(rappel+précision)

4.2 Recherche

Dans cette phase, nous avons créé 5 requêtes reparties en 05 classes :

requete1: The agent client is a lazy person who does request the price of a car, but didn't tell us its exact needs.Maybe the client is interested in a cheep/expensive/fast car.

requete2: The client is interested to know about food that is available in a certain grocery store. It wants to actually buy that food from the grocery shop.

Requete3: The client wants to find a title of comedy film.

Requete4: The client wants to travel from Frankfurt to Berlin, that's why it puts a request to find a map to locate a route from Frankfurt to Berlin.

Requete5: The client is interested to know about the funding provided by a government for development of a missile (especially ballistic missiles).

Les deux tableaux suivants indiquent les résultats obtenus, en utilisant uniquement les vecteurs de fréquences de termes (sans inputs & outputs):

service	Score	Pertinence
S1	0,476731270551682	Oui
S2	0,476731270551682	Oui
S3	0,301511347293854	Oui
S4	0,476731270551682	Non
S5	0,402015119791031	Oui
S6	0,319801062345505	Oui
S7	0,369274437427521	Oui
S8	0,369274437427521	Oui

Table1 : score de la recherche en utilisant cosine (pour la cinquième requête)

La table 2 indique l'influence du taux de matching θ2 (ie le seuil au delà du quel la réponse est considérée comme pertinente), sur les critères de rappel et de précision (pour l'ensemble des requêtes).

Requêtes	Taux de matching θ2	rappel	précision	F-mesure
Requête 1	≥27%	100%	83%	90,5%
	≥30%	83%	83%	83%
Requête 2	≥27%	100%	100%	100%
	≥30%	100%	100%	100%
Requête 3	≥27%	100%	100%	100%
	≥30%	75%	100%	85,71%
Requête 4	≥27%	100%	100%	100%
	≥30%	100%	100%	100%
Requête 5	≥27%	100%	90%	94,73%
	≥30%	100%	90%	94,73%

Table2: influence du taux de matching sur la qualité des résultats

De façon générale, nous remarquons que les scores de recherche (matching) sont tous compris entre 0 et 0,60 (à cause de la taille reduite de la requete et la nature syntaxique de la mesure cosine), plus le taux de matching est petit, plus le rappel est bon et plus la précision est mauvaise et vice versa. Nous remarquons que la bonne valeur du taux est située autours du 30%. Ces résultats peuvent être améliorés en combinant des mesures de similarité semantique, ou bien en utilisant des concepts au lieu des termes.

Les deux tableaux suivants montrent les résultats obtenus en utilisant les concepts d' inputs/outputs, et la mesure cosine (sans vecteurs de fréquences de termes).

service	**Score**	**Pertinence**
S1	0,59	Oui
S2	0,61	Oui
S3	-	Oui
S4	-	Non
S5	0,78	Oui
S6	0,75	Oui
S7	0,63	Oui
S8	0,82	Oui

Table 3 : score de la recherche en utilisant cosine (pour la cinquième requête)

requêtes	**rappel**	**précision**	**F- mesure**
Requête 1	100%	100%	100%
Requête 2	100%	57,14%	72,75%
Requête 3	100%	100%	100%
Requête 4	100%	100%	100%
Requête 5	100%	85%	91%

Table4 : Résultats obtenus en utilisant les inputs/outputs (Taux de matching $\theta 1 = 0,5$)

Nous remarquons que l'utilisation des concepts d'inputs / outputs augmente le niveau de la précision, nous remarquons aussi que les scores de comparaison obtenus sont assez élevés, ce qui permet de lever l'ambiguïté et d'améliorer la

précision. (pour cette raison nous avons pris un taux de matching θ1 assez élevé (0,5)).

Donc nous montrons expérimentalement que l'indexation sémantique des entrées/sorties des services web améliore nettement les performances de la recherche, et peut compléter les lacunes de la recherche d'information classique, et même les lacunes des approches de recherches fondées sur les logiques de description, comme on le confirme dans [15].

Conclusion

Nous avons proposé dans ce papier, une approche de recherche des services web basée sur les techniques de recherche d'information (les fréquences des termes, cosine), et les ontologies.

Le processus de recherche se déroule en deux phases, nous comparons en premier lieu, le vecteur de la requête avec les différents représentants des classes de services, après nous comparons, toujours la même requête avec les membres des classes retenues lors de la phase précédente. Cette comparaison est basée sur la mesure de similarité cosine et les descripteurs du document OWLS.

Nous pouvons citer plusieurs perspectives à ce travail, en premier lieu nous suggérons la comparaison des performances d'un ensemble de mesures de similarité (syntaxiques et sémantiques), dans la recherche, nous proposons aussi la prise en charge d'un autre ensemble de descripteurs d'OWLS dans la modélisation.

Références

[1] Banaei-Kashani, F.; Chen, C.-C.; and Shahabi., C. 2004. Wspds: Web services peer-to-peer discovery service. In Proceedings of International Symposium on Web Services and Applications (ISWS).

[2] Bansal, S., and Vidal, J. 2003. Matchmaking of web services based on the daml-s service model. In Proceedings of Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Melbourne, Australia.

[3] Benatallah, B,. Hacid, M-S Rey, C. & Toumani, F. Semantic Reasoning for Web Services Discovery, WWW Workshop on E-Services and the Semantic Web, Budapest, Hungary. (2003).

[4] Bernstein, A., and Klein, M. 2004. Towards high-precision service retrieval. In IEEE Internet Computing, 8(1):30-36.

[5] Cohen, W.; Ravikumar, P.; and Fienberg, S. 2003. A comparison of string distance metrics for name-matching tasks. In Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03). DBLP at http://dblp.uni-trier.de.

[6] Colucci, S.; Noia, T. D.; Sciascio, E. D.; Donini, F.; and Mongiello, M. 2004. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In Proc. 6th Int Conference on Electronic Commerce (ICEC 2004). ACM Press.

[7] Constantinescu, I., and Faltings, B. 2003. Efficient matchmaking and directory services. In Proceedings of IEEE/WIC International Conference on Web Intelligence.

[8] Curbera, F., Duftler, M. Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana ,S. .Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 6(2). (2002).

[9] Curbera, F., Nagy, W., and Weerawarana, S.. "Web Services: Why and How." Workshop on Object-Oriented Web Services – OOPSLA, Tampa, Florida, USA. (2001).

[10] 10 . Dean, M., (ed). OWL-S: Semantic Markup for Web Services. Version 1.0 (2004).

[11] Hadjila, F., M, Malki. Les services web sémantiques: Une approche de découverte basée sur les agents CIIA Saida, Algérie. (2006).

[12] Horrocks, I.; Patel-Schneider, P.; and van Harmelen, F. 2004. From shiq and rdf to owl: The making of a web ontology language. Web Semantics, 1(1), Elsevier.

[13] Keller, U.; Lara, R.; Polleres, A.; and Fensel, D. 2005. Automatic location of services. In Proceedings of European Semantic Web Conference (ESWC), Springer, LNAI 3532.

[14] Klein, M., and Koenig-Ries, B. 2004. Coupled signature and specification matching for automatic service binding. In Proceedings of European Conference on Web Services, Springer, LNAI, 183-197.

[15] Klusch, M.;Fries, B.; Mahboob, K.; Sycara, K.; OWLS-MX: Hybrid OWL-S Service Matchmaking, American Association for Artificial Intelligence. 2005 (www.aaai.org)

[16] Li, L., and Horrock, I. 2003. A software framework for matchmaking based on semantic web technology. In Proc. 12th Int World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW 2003).

[17] Mandell, D., and McIllraith, S. 2003. A bottom-up approach to automating web service discovery, customization, and semantic translation. In Proc. 12th Int Conference on the World Wide Web (WWW 2003). ACM Press.

[18] Sycara, K.; Klusch, M.; Widoff, S.; and Lu, J. 2002. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Autonomous Agents and Multi-Agent Systems, 5(2), Kluwer.

[19] Sycara, K.; Paolucci, M.; Anolekar, A.; and Srinivasan, N. 2003. Automated discovery, interaction and composition of semantic web services. Web Semantics, 1(1), Elsevier.

[20] Verma, K.; Sivashanmugam, K.; Sheth, A.; Patil, A.; Oundhakar, S.; and Miller., J. 2004. Meteor-s wsdi: A scalable infrastructure of registries for semantic publication and discovery of web services. Information Technology and Management.