Création de surcouche de documents hypertextes et traitement du langage naturel

David Nadeau
École d'ingénierie et de technologie de l'information
Université d'Ottawa
dnadeau@site.uottawa.ca

Résumé

Cet article présente une extension aux algorithmes de création de surcouche de documents hypertextuels. Il s'agit de diversifier la granularité de l'information qu'il est possible de capturer en utilisant des techniques de traitement du langage naturel. Une surcouche de document Web (*web page wrapper*) est une vue sur des nœuds HTML contenant une information donnée et désirée. Par exemple, dans une manchette de journal, une surcouche peut baliser le nom de l'auteur, la date ou même toutes les références à un lieu ou a une compagnie quelconque. Nous avons étendu le fonctionnement d'un algorithme de création de surcouchage afin de dépasser la limite des nœuds HTML et d'extraire de l'information du contenu textuel qui s'y retrouve. Nous appliquons cette technique à la création automatique de lexiques (liste de mots).

1 Introduction

La création de surcouche de document hypertexte est un concept abstrait. Certains auteurs expliquent qu'un tel algorithme fait la réécriture de la page [Cohen et Fan, 1999], que l'information est simplement extraite sous forme d'une liste [Etzioni et al. 2004] ou alors qu'un ensemble de règles (ou grammaire) permet de discriminer les bons nœuds HTML du reste [Hong et Clark, 2004]. Dans tous les cas, la surcouche opère au niveau des nœuds HTML et utilise l'apprentissage machine pour induire un modèle qui, une fois appliqué aux documents hypertexte, permet d'isoler les nœuds contenant l'information prisée. Un exemple intéressant est introduit par Etzioni et al. [2004], qui combine l'induction de surcouche et la recherche d'information (*information retrieval*) pour générer automatiquement une liste de plus de 20,000 villes à partir de seulement 4 noms de villes fournies par un humain et l'analyse de toutes les pages qui les contiennent toutes en co-occurrence via un engin de recherche.

La principale limitation de ce genre d'algorithme est que, dans ce cas bien précis, tous les noms de villes doivent être contenus dans un nœud HTML, sans autre bruit. Par exemple, un seul nom de ville peut être correctement extrait du code suivant:

```
Bienvenue à <b>Gatineau</b>!
<p>Bienvenue à Montréal!</p>
```

La nom de ville "Gatineau" est effectivement circonscrit dans le nœud `<b>…</b>` alors que le nom de ville "Montréal" est dans un contexte textuel plus large au sein du nœud `<p>…</p>`.

Dans cet article, nous présentons une extension de l'algorithme de création de surcouches de Cohen et Fan [1999]. Nous proposons un système basé sur la programmation logique inductive qui permet d'identifier des patrons à l'intérieur des nœuds HTML et ainsi extraire de l'information à un niveau de granularité plus fin. Nous nous sommes inspirés des travaux en extraction d'information à partir de documents textuels pour dépasser la contrainte du nœud tout en conservant

l'information structurelle qu'offre le document hypertexte.

L'article est divisé comme suit : la section 2 présente la littérature pertinente à la présente recherche, la section 3 introduit l'algorithme de surcouchage de documents hypertexte, la section 4 définit le problème auquel nous faisons face, la section 5 détaille notre approche pour l'identification de patrons à l'intérieur des nœuds html et, finalement, la section 6 présente une évaluation du système résultant au niveau du gain en rappel d'information pour la tâche proposée par Etzioni et al. [2004].

2 Travaux connexes

Ce travail se situe à l'intersection de deux champs de recherche. Il y a d'abord les algorithmes de création de surcouches de documents hypertextes (*web page wrapper*) qui consistent à cibler - grâce à l'analyse de la position, du voisinage et du contenu - les nœuds html contenant de l'information de choix. Il y a aussi l'extraction d'information (*information extraction*) plus classique qui opère sur des documents textuels et qui consiste à remplir des champs prédéfinis (e.g. : auteur, date, compagnie, lieu, etc.) grâce à des systèmes de règles ou des automates probabilistes.

Le surcouchage de document hypertexte, au plus simple, consiste à déterminer des règles qui identifient des nœuds HTML pertinents pour une tâche donnée. Notamment, le méta moteur de recherche Copernic de la fin des années 90 utilisait des règles de surcouchage induites par des humains pour identifier les différentes parties d'un résultat de moteur recherche. Par exemple une règle pouvait être, « `le sixième champ textuel en caractère gras à partir du haut de la page est le nombre de résultats du moteur de recherche Yahoo!` ».

Avec l'apprentissage automatique, ce genre de règles peut être induit en généralisant sur une banque d'exemples annotés. Ashish et Knoblock [1997] proposent un algorithme d'induction opérant sur des documents hypertexts semi-structurés. Ils utilisent des hypothèses de départ, d'où la contrainte « semi-structurée », comme le fait qu'un titre se retrouve dans un nœud `<h>…</h>` ou qu'une indentation marque un changement de section. Ils apprennent alors à reconnaître les sections d'une page en utilisant la structure connue pour délimiter le contenu.

Kushmerick [1997] va plus loin en présentant un algorithme d'induction de surcouches applicable à un ensemble de pages provenant du même site web, sans hypothèse de départ. Par exemple, étant donné des exemples de pages de résultat d'un moteur de recherche donnée, le surcouchage de pages subséquentes est automatiquement induit. Chaque moteur de recherche requiert ici un entraînement particulier. Kushmerick définit la surcouche par l'entourage de l'information. Par exemple, ce qui discrimine le nombre de résultats d'un moteur est qu'il est précédé par la séquence « `<font size=1><b>` » et suivi de « `</b> hits` »

Cohen et Fan [1999] présente finalement une généralisation des travaux précédents en mettant au point un algorithme qui permet d'apprendre à extraire de l'information au niveau des nœuds HTML, indépendamment des sites web. Cela signifie, pour reprendre encore une fois l'exemple du moteur de recherche, qu'un seul algorithme permet d'identifier l'information voulue et ce, indépendamment du moteur. La surcouche est ici définie au niveau des nœuds HTML. Par exemple, « `le nombre de résultats du moteur de recherche est un nœud près du début de la page, tout juste après une ligne horizontale, contenant une chaîne de caractères en caractères gras de longueur inférieure à 10` ». C'est cet algorithme qui sert de base à notre travail et nous le présentons en détails à la section 3.

L'extraction d'information est plus couramment appliquée aux documents textuels non structurés qu'aux documents hypertextes. Typiquement, à la façon des compétitions comme MUC (*Message Understanding Conference*), un article journalistique d'un domaine particulier (e.g. : le terrorisme) est donné en entrée et des informations particulières (e.g. : victime(s), lieu, etc.) doivent être extraites en sortie. Plusieurs méthodes performantes ont été proposées dans une littérature abondante mais le *modus operandi* tourne le plus souvent autour des méthodes d'apprentissage de règles [Soderland, 1999], [Califf et Mooney, 2003] et les modèles statistiques [Bikel et al., 1997]

comme le modèle de Markov caché (*Hidden Markov Model*).

Soderland [1999] s'intéresse à l'extraction d'information comme le prix et le lieu dans des annonces classées d'immobilier. L'apprentissage des règles, qui s'apparentent aux expressions régulières, consiste à analyser plusieurs séquences de mots qui contiennent l'information à extraire et d'en déduire une généralisation permettant de couvrir le plus de cas possible mais de conserver une spécialisation suffisante pour ne pas récolter trop de fausses informations. Comme c'est souvent le cas, Soderland permet certaines généralisations d'ordre syntaxiques (parties du discours, racine de mots) et sémantiques (entités nommées, classes de mots prédéfinies dans le domaine). Le système utilise une logique de couverture, c'est-à-dire qu'à une itération donnée, la règle la plus générale possible est créée et les exemples couverts sont retirés pour la suite.

Califf et Mooney [2003] présentent le système RAPIER un système qui apprend à extraire de l'information de documents textuels pour satisfaire des champs prédéfinis. L'intérêt de RAPIER est l'utilisation de la programmation logique inductive qui opère sur des prédicats (e.g. : le mot X précède le lemme Y) et n'impose pas de limites quand à la taille ou suite d'opérations définissant une information. Le système utilise un logique de compression, c'est-à-dire que chaque exemple d'entraînement est décrit en terme de la liste de prédicats la plus spécifique possible. L'ensemble des règles est alors comprimé en cherchant la généralisation qui couvre le plus de règles spécifiques. Un juste milieu entre l'informativité (précision et étendue de la couverture) et la complexité des prédicats est recherché. Comme dans [Soderland, 1999], la généralisation est permise sur la racine des mots, certaines classes sémantiques mais aussi les parties du discours.

De son coté, Bikel et al. [1997] présentent un automate statistique à états finis inspiré du modèle de Markov caché. La tâche visée est l'extraction d'entités nommées (noms de personnes, de lieux, de sociétés). L'entourage de chaque entité est analysé et défini en terme d'attributs. Plus particulièrement, chaque mot est décrit par sa teneur en chiffres, sa casse et sa position dans la phrase. Chaque état de l'automate est en fait la description d'un bigram de mots qui précède ou suit l'entité en question et chaque transition se voit assigner une probabilité, selon les observations faites sur l'ensemble d'entraînement. On peut s'attendre, par exemple, qu'un mot précédé de « Mr » et de la ponctuation « . » ait une forte probabilité d'être un nom de personne.

La section qui suit présente notre approche au surcouchage de documents hypertextes, qui est une reproduction fidèle du modèle de Cohen et Fan [1999] auquel nous avons ajouté de nouveaux attributs. Plus loin, la section 5 présente la façon dont nous traitons l'information textuelle comprise à l'intérieur des nœuds HTML. Nous avons opté pour le modèle d'apprentissage de règles RAPIER de Califf et Mooney [2003] auquel nous avons apporté des simplifications majeures au niveau de la recherche dans l'espace de règles étant donné le contexte particulier dans lequel nous l'appliquons.

La combinaison de ces deux techniques - surcouchage et traitement du langage naturel - est la principale contribution de cet article.

3 Algorithme de surcouchage

Créer une surcouche pour un document hypertexte consiste à identifier les noeuds HTML qui contiennent l'information désirée. Un document hypertexte peut être vu comme un arbre pour lequel le nœud `<html>` est à la racine. Pour les fins de l'apprentissage machine, les nœuds qui contiennent l'information visée sont des exemples positifs et tout le reste de l'arborescence contient les exemples négatifs. Par exemple, dans le code HTML suivant, le nom de ville Ottawa est désiré et constitue un exemple positif. Le nœud `<a>…</a>`, en caractère gras, est visé :

```
<tr>
<td>Day5</td>
<td><img src="bullet.gif">
<a href="vacation.htm">
Ottawa
</a>
</td>
<td>Ottawa, Museum of Civili-
zation : Morning drive to
Canada's capital city, Ot-
tawa. This afternoon visit
the Canadian Museum of Civi-
lization…</td>
</tr>
```

L'identification des noeuds positifs peut être formulé comme un problème de classification (noeuds à extraire vs nœuds à ne pas extraire). Dix-huit attributs permettent de décrire un nœud HTML au sein de l'arborescence. Dans [Cohen et Fan, 1999], l'algorithme d'apprentissage RIPPER est utilisé pour classer les noeuds. Par rapport à l'approche originale, nous utilisons deux attributs nouveaux et nous avons omis trois attributs nous semblant redondants.

L'ajout de deux attributs ayant des pouvoirs prédictifs élevés nous apparaît une contribution intéressante à l'approche de Cohen et fan. Ces attributs sont le numéro de la rangée et de la colonne dans la table immédiate à laquelle le nœud appartient. Nous n'avons toutefois pas évalué l'apport exact de ces deux attributs dans le présent travail.

Voici la description de tous les attributs qui sont soit numériques (valeur réelle), soit nominaux (choix parmi un ensemble de valeurs prédéfinies).

Étiquette du noeud: *nominal* {div, td, img, p, a, …}[1]

Longueur du texte: *numérique*

Longueur du texte sans les espaces: *numérique*

Longueur récursive du texte: *numérique*

Longueur récursive du texte sans les espaces: *numérique*

Profondeur: *numérique*

Profondeur normalisée[2]: *numérique*

Nombre d'enfants: *numérique*

Nombre d'enfants normalisé: *numérique*

Nombre de voisins: *numérique*

Nombre de voisins normalisé: *numérique*

Étiquette du noeud parent: *nominal* {div, td, img, p, a, …}

Nombre de préfixes: *numérique*

Nombre de préfixes normalisés: *numérique*

Nombre de suffixes: *numérique*

Nombre de suffixes normalisé: *numérique*

Numéro de rangé dans la table immédiate: *numérique*[3]

Numéro de colonne dans la table immédiate: *numérique*

Classe: {Positif, Négatif}

Voici la description d'un noeud typique, selon cette représentation:

```
a,6,0,0,4002,26,0.684211,8,0.
222222,1,0.027778,td,104,0.51
4851,0,0,2,1, Positif
```

Ce noeud est un exemple positif. L'étiquette du nœud est « a » (il s'agit d'un hyperlien) et le nœud parent est « td » (une cellule d'une table). Le nœud contient six caractères et il n'y a pas de caractères dans les nœuds enfants (longueur récursive). Le nombre de préfixes, entre autres, est 104, ce qui signifie que 104 autres nœuds partagent le même préfixe (e.g.: html>body>table>tr>td). Les nouveaux attributs indiquent ici que le nœud est à la rangée 2 et à la colonne 1 de la table immédiate.

[1] Cette énumération devrait contenir toutes les étiquettes possibles pour un nœud.

[2] Référez à [Cohen and fan, 1999] pour la description des attributs et les détails de la normalisation.

[3] Le numéro de rangée et de colonnes sont de nouveaux attributs. Si jamais un nœud n'est pas à l'intérieur d'une table, ces valeurs sont nulles.

4 Définition du problème

La force de l'algorithme de surcouchage - et à la fois sa principale limitation - c'est qu'il exploite la structure du document hypertexte. Plusieurs travaux en extraction d'information démontrent que cette approche donne d'excellents résultats pour différentes tâches, par exemple [Craven *et al*. 2000], [Etzioni *et al*. 2004]. Cependant, si la granularité de l'information est inférieure au nœud, la différenciation n'est pas possible.

Dans ce travail, nous proposons pour la première fois une technique permettant d'aller extraire de l'information à l'intérieur des nœuds où se trouve du texte brut. Un travail qui s'apparente à cette idée est proposé par Freitag et Kushmerick [2000] et permet d'atteindre cette même granularité dans un document hypertexte. Cependant, leur approche est une application de l'extraction d'information classique et consiste à traiter le document comme du texte brut. Les nœuds HTML sont alors vus comme une suite de lettres et de ponctuations pouvant être utilisés pour former des règles. Cette approche, contrairement à la notre, perd toute notion d'arborescence et de structure hypertexte. Dans notre travail, nous profitons de la définition d'un problème d'extraction précis pour en arriver à appliquer notre technique efficacement. À ce point, il importe de définir ce problème, tel que présenté par Etzioni *et al*. [2004].

Etzioni *et al.* [2004] ont démontré que la construction de lexiques par surcouchage de documents hypertexte est une technique prometteuse qui surpasse l'apprentissage de règles classique. Spécifiquement, étant donné *k* noms de villes, il s'agit de retrouver des documents qui les contiennent tous. Raisonnablement, si un document contient *k* noms de villes, il y a fort à parier qu'il en contient plus de *k*. Il faut alors identifier si les noms de villes sont contenus dans des nœuds et apprendre à classer ces nœuds grâce au surcouchage. Le but d'une telle tâche est de créer un lexique de villes. En n'ayant que k noms de villes au départ, il est possible d'en détecter de nouveaux dans les documents. En échantillonnant *k* nouvelles villes parmi celles identifiées par surcouchage, il est possible d'itérer à nouveau. Etzioni *et al*. ont démontré qu'il est ainsi possible de créer un lexique de plusieurs dizaines de milliers de noms de villes.

Le problème évident de cette approche est que les noms de villes ne sont pas nécessairement isolés dans un nœud HTML. Même que c'est l'inverse dans la majorité des cas. Par exemple, il est très fréquent de rencontrer des segments comme « `New York Hotels` », « `Toronto restaurants` », ou simplement « `Quebec city` ». En analysant les 100 premiers résultats de recherche de Google pour les mots « Montreal », « Mexico », « Denver » et « New York », on dénombre 3 pages dans lesquelles les 4 villes sont strictement incluses dans des nœuds alors que 9 pages présentent les noms de villes dans des contextes textuels plus large. Nous posons l'hypothèse qu'en traitant ces dernières, nous dépasserons le taux de rappel d'Etzioni *et al.* et retrouverons plus de noms de villes en moins d'itérations.

5 Traitement de l'information textuelle intra-nœuds

Nous proposons une technique pour arriver à diviser les entités de leur contexte au sein d'un nœud HTML (e.g. : séparer « New York » de « Hotel ») dans le cas général. C'est-à-dire que la méthode ne s'applique pas seulement aux noms de villes mais à tout autre entité faisant l'objet du même problème, par exemple séparer le nom du constructeur de voiture dans « Ford Focus ZX5» ou la profession dans « le peintre Paul-Émile Borduas ».

Nous profitons bien entendu d'une connaissance de départ, soit les *k* entités choisies par un humain. Étant donné un document web où les *k* entités apparaissant dans des contextes textuels larges au sein de nœuds HTML, nous utilisons les *k* contextes pour entraîner un modèle d'apprentissage de règles similaire à RAPIER de Califf et Mooney [2003]. Cependant, nous sommes en mesure d'y faire une simplification importante puisque nous savons a priori que nous avons *k* exemples d'entraînement. Qui plus est, nous limitons de façon empirique la taille des contextes (entité exclue) à 50 caractères[4], vu la nature même du problème. Les fonctions de sélection d'exemples aléatoires ou de recherche dans un espace d'exemples peuvent être ignorées puisque, dans le

[4] Bien que l'impact d'élargir le contexte à 100 ou 200 caractères a des conséquences négligeables sur la performance.

pire des cas, disons que les 50 caractères forment 25 mots et 25 espaces, il y aura $O^{25 \bullet \binom{k}{2}}$ tests de généralisation à effectuer, ce qui est loin d'être un problème de performance. En effet, à la limite, chaque paire de mots parmi les *k* contextes peuvent être soumis à un test de généralisation. Bref, le problème particulier sur lequel nous travaillons nous permet des simplifications importantes.

RAPIER [Califf et Mooney, 2003] est un système de génération de règles basé sur la programmation logique inductive. Logique parce qu'il définit les séquences de mots du contexte par le prédicat "X est précédé de Y" ou "X est suivi de Y" (ce qui rend l'algorithm bidirectionnel). Inductif parce qu'il déploie une stratégie de compression ayant comme point de départ les définitions les plus spécifiques possibles, graduellement remplacées par la généralisation la plus spécifique possible.

La séquence de généralisation permise, pour deux mots, est leur forme canonique (minuscule, normalisé, etc.) puis leur partie du discours[5] mais seulement parmi les classes adverbe, conjonction, déterminant, nombre, clitique, pronom et nom propre.

La figure 1 présente visuellement un exemple d'application de l'algorithme sur les *k*=4 contextes suivants, où chaque mot est définit par un tuple $\langle O, C, P \rangle$: forme (O)riginale, forme (C)anonique et (P)artie du discours. Le jeton spécial ε signifie que le mot peut être absent du contexte. Il n'y a bien sûr aucun problème à isoler les noms de villes puisqu'elles sont connues au moment de l'entraînement.

```
1) Montreal <Hôtels, hotels, N>

2) Mexico <city, city, N> <Ho-
tels, hotels, N>

3) Denver <hotels, hotels, N>

4) New York <City, city, N>
```

[5] Les parties du discours sont déterminées grâce au logiciel libre Balie (http://balie.sourceforge.net)

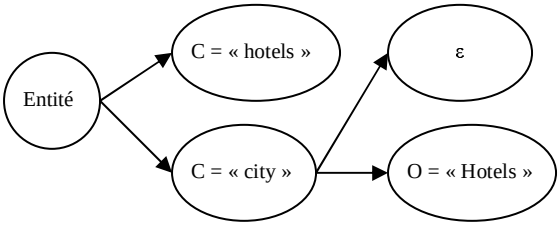

Figure 1: exemple d'application de l'algorithme.

L'utilisation des règles créées à la façon de RAPIER s'utilisent de façon similaire à des clauses de grammaires formelles. Il s'agit donc de considérer un nouveau contexte et de vérifier s'il est possible de l'accepter dans le langage défini par les règles. Si c'est le cas, la section acceptée est retirée du contenu textuel et on obtient en sortie l'entité épurée.

Utilisé conjointement avec le surcouchage de documents hypertextes, la création de règles permet d'apprendre les règles les plus spécifiques (priorité à la précision) qui décrivent le contenu textuel dans le voisinage des entitiés. Les nouveaux noeuds qui sont classées positifs sont alors analysés pour en extraire les entités épurées, si le système de règles est satisfait.

6 Évaluation

La technique que nous proposons a pour but d'augmenter le rappel de l'extraction d'information fait à la façon et Etzioni *et al.* [2004]. Nous avons relâché la contrainte d'inclusion stricte d'une entité dans un noeud HTML en déployant une technique de traitement du langage naturel. Dans cette section, nous évaluons cette contribution.

Pour un problème comme celui des villes, la technique de base est intuitivement efficace puisqu'il y a plusieures listes de villes auxquelles il est possible d'appliquer l'algorithme de surcouchage seul. Cependant, pour un problème comme celui des marques de voitures, on peut imaginer qu'il y aura beaucoup plus de listes contenant aussi le nom des modèles ou les années de fabrications, ce qui est un bruit supplémentaire.

Pour le problème des villes, le point de comparaison que nous avons est le suivant. Etzioni

et al. [2004] mentionnent que leur technique de surcouchage permet de retrouver ~20,000 noms de villes, en itérant un nombre inconnu de fois compris en 5,000 et 10,000, avec un ensemble de départ (inconnu) de taille k=4. Chaque itération consiste à échantillonner k entitiés au hasard dans la liste et à les utiliser pour faire des requêtes à un moteur de recherche. Etzioni calcule que l'ensemble final extrait est précis à 90%. Pour en arriver à ce nombre, un échantillon est évalué manuellement et la précision est approximée sur l'ensemble.

Reproduire cette expérience pose plusieurs problèmes. D'abord certaines information sont inconnues ou variables (sélection aléatoire) mais, surtout, l'accès au moteur de recherche Google à des fins de recherche est limité à 1,000 requêtes par jour. Sachant que chaque requête retourne 10 résultats (limite de l'interface de programmation) et que nous analysons 100 résultats par itération, nous sommes limités à seulement 100 itérations par jour[6]. Il est en effet impensable d'étaler l'exécution du programme sur 50 jours pour atteindre 5,000 itérations.

Nous avons utilisé, comme échantillon de départ, les noms de villes `"Ottawa", "Toronto", "Paris"` et `"Dallas"`. Nous posons l'hypothèse que l'algortihme de base (sans traitement intra-noeud) permet d'obtenir un rappel équivalent à l'équipe d'Etzioni (~20,000 villes). Les résultats sur 100 itérations laissent présager que cette hypothèse est conservatrice puisque nous obtenons rapidement un rappel de 9,706 villes.

Nous avons donc exécuté notre algorithme avec et sans le traitement intra-noeuds. La figure 2 montre le rappel en fonction du numéro d'itération.

Le traitement intra-noeud procure une augmentation de près de 76% par rapport au surcouchage seul.

Nous avons aussi évalué la précision de l'algorithme en vérifiant un échantillon de 100 villes, choisies au hasard. La table 1 montre cette précision pour la technique de base et l'extension intra-noeud.

[6] Une itération prend typiquement moins d'une minute à exécuter. La majeur partie du temps est passée à récupérer les 100 documents hypertexte sur le web.

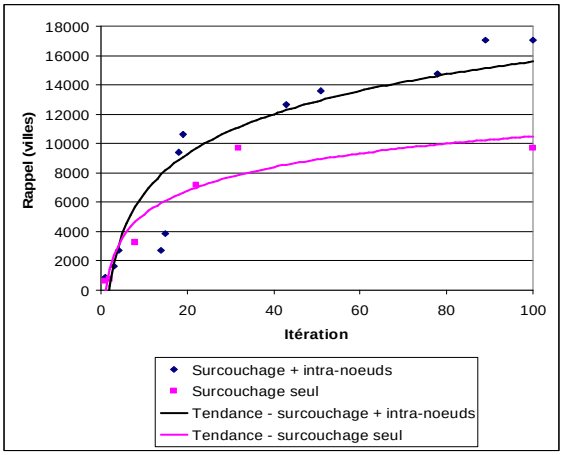

Figure 2: Résultats pour le problème des villes.

La perte de précision introduite par la méthode intra-noeud semble tout à fait raisonnable puisque la précision reste au delà de 90%, ce qui est le minimum proposé par Etzioni *et al.* Les erreurs typiques sont des expressions d'entités géographiques (e.g.: "country club estates", "elk river", "grand canyon"), des éléments de formatage (e.g.: lettres de l'alphabet) ainsi que des glissements de sens (e.g.: provinces, états, pays).

Stratégie	***Précision***
Surcouchage seul	98%
Surcouchage + intra-noeuds	97%

Table 1: Précision sur un échantillon de 100 villes.

Le traitement de l'information intra-noeud procure donc une augmentation du rappel très importante sans pour autant compromettre la précision. En effet, le rappel atteint 17,065 noms de villes après seulement 100 itérations alors que le surcouchage seul ne dépasse pas 9,706. Ce résultat se compare même avantageusement bien aux travaux d'Etzioni *et al.* [2004].

Nous avons fait le même expérience pour extraire des noms de voitures, l'échantillon de départ étant composé des noms `"volkswagen golf", "mazda miata", "ford focus"` et `"nissan maxima"`. Le rappel atteint 2,689 après 100 itérations en utilisant le surcouchage seul. Il atteint cependant 3,151 en ajoutant le traitement intra-nœud. Le problème des noms de voitures semble particulièrement intéressant pour notre méthode puisque ceux-ci sont très souvent accompagnés d'un nom de modèle ou d'une année de fabrication, bien qu'il n'y ait pas de généralisa-

tion possible dans le premier cas. La précision calculée, pour les listes de voitures, frôle 100% dans les deux cas. Les erreurs typiques sont des expressions reliées au commerce des voitures (e.g. : « car rental », « best car price »).

7 Conclusion

Dans cet article, nous avons présenté une extension originale au surcouchage de documents hypertextes basée sur le traitement du langage naturel. Nous avons appliqué cette techniques au problème de création automatique de lexique de noms de villes.

Exprimé autrement, il s'agit d'une combinaison de techniques d'extraction d'information qui exploite à la fois la structure des documents et le texte brut.

Les résultats obtenus, sur la tâche d'extraction de noms de villes, démontrent qu'il est ainsi possible d'obtenir un taux de rappel significativement plus élevé comparativement à l'approche du surcouchage seule sans compromettre la précision de l'extraction. La contrainte imposée par la méthode de base est en effet très stricte et peu de documents web sont des candidats valides. Grâce à l'extension proposée, le nombre de pages candidates est augmenté d'un facteur important, ce qui est crucial dans le cas d'entitées plus rare que les villes. Nous n'avons pas évalué ce facteur précisément mais cette mesure est au menu de nos travaux à venir, tout comme l'application de cette technique à d'autres types d'entités.

L'extraction de lexiques a des applications en création automatique d'ontologies (objets au même niveau dans la hiérarchie), en recherche d'information (e.g.: expansion de requêtes) et en support non-supervisé à l'extraction d'information traditionnelle (découverte de l'extension de classes sémantiques de mots).

Le code source de l'algorithme de surcouchage et de l'extension pour le traitement du langage naturel utilisé dans le cadre de ce travail est disponible sous forme de logiciel libre à l'adresse `http://balie.sourceforge.net`.

References

[Ashish et Knoblock, 1997] Ashish, N. and Knoblock, C., Wrapper Generation for Semi-structured Internet Sources, *Workshop on Management of Semistructured Data*, 1997.

[Bikel *et al*. 1997] Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R., Nymble: a high-performance learning name-finder, *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1997.

[Califf et Mooney, 2003] Califf, M. E. and Mooney, R. J., Bottom-up relational learning of pattern matching rules for information extraction, *Journal of Machine Learning Research*, vol. 4. 2003.

[Cohen et Fan, 1999] Cohen, W. and Fan, W., Learning Page-Independent Heuristics for Extracting Data from Web Page, *in WWW-99*, 1999.

[Craven et al. 2000] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S, Learning to construct knowledge bases from the world wide web, Artificial Intelligence 118:69–113, 2000.

[Etzioni et al., 2004] Etzioni, O., Cafarella, M., Downey, D., Popescu A.-M., Shaked, T., Soderland, S., Weld, D.S., and Yates A., Methods for Domain-Independent Information Extraction from the Web: An Experimental Comparison, American Association for Artificial Intelligence (AAAI), 2004.

[Freitag et Kushmerick, 2000] Freitag, D. and Kushmerick, N., Boosted wrapper induction, *In Proc. of the 17th National Conference on Artificial Intelligence AAAI-2000*, 2000.

[Hong et Clark, 2004] Hong, T. W. and Clark, K. L., Towards a Universal Web Wrapper. *17th International FLAIRS Conference*, 2004.

[Kushmerick, 1997] Kushmerick, N. *Wrapper Induction for Information Extraction*, Ph.D. Dissertation, Department of Computer Science & Engineering, University of Washington, 1997.

[Soderland, 1999] Soderland, S., Learning Information Extraction Rules for Semi-Structured and Free Text, *Machine Learning*, 34(1-3), 1999.