Fadjimba KOUYATE **11312697**
Amine ADIB **11005061**

SMExtract : un outil d'extraction de relations entre entités spatiales

Résumé

Une grande quantité de données cartographiques existent dans de nombreuses applications. Cela permet d'avoir un nombre considérable de relations entre entités, comme par exemple la phrase *" Le centre commercial La Part-Dieu se situe à Lyon"* qui montre une relation de localisation entre deux entités (*Part-Dieu* et *Lyon*).
Mais ces relations sont en général des relations topographiques. Pourtant il existe bien des relations non topographiques dans des documents textuels qui lient les entités comme cette phrase *" Le centre commercial Confluence est plus beau que celui de la Part-Dieu"* qui montre une relation d'esthétique entre les deux entités.

Dans ce TER nous proposons une application nommée SMExtract, qui se sert d'outil existant pour extraire des entités dans des documents textuels. Et qui, à partir d'algorithme flexible, détecte les types de relations non topographiques existants entre ces entités.

Mots-clés : Entité spatiale, relation non topographique, SMExtract, analyse linguistique, outil d'extraction

1 - Introduction :

De nos jours, les applications de gestion de données cartographiques, comme l'aménagement du territoire, le suivi d'une personne ou encore la localisation d'un lieu sont en pleine expansion.
Malgré cela il existe peu d'informations sur les relations entre entités spatiales, en particulier pour des types de relations non topographiques. Ces informations existent pourtant, mais apparaissent sous forme textuelle et sont donc difficilement exploitables.

Comment extraire ces entités spatiales ainsi que les relations non topographiques qui les lient?

Il existe plusieurs outils qui offrent des services d'annotation sémantique et qui sont de plus en plus utilisés par de nombreuses applications [1,3]. Ces outils existants détectent dans des textes des entités et découvrent le type de relation qui lie ces entités. Mais ils sont souvent génériques et ne se focalisent que sur du spatial.

Nous nous intéressons particulièrement à la détection de relations non topographiques entre entités spatiales. Nous utiliserons l'un des outils existant afin de détecter les entités spatiales avant de nous focaliser sur les types de relations qui les lient. L'approche que nous proposons consiste d'abord à analyser ces relations et d'en extraire ensuite celles qui respectent certaines contraintes syntaxiques et lexicales que nous avons établies.

Après une brève description des travaux existants (Section 2), nous présentons un algorithme pour détecter les types de relations entre des entités spatiales (Section 3). Par ailleurs nous décrivons notre outil SMExtract qui implémente l'algorithme mentionné ci-dessus et qui aide les utilisateurs à construire une base d'apprentissage pour le spatial en validant les types de relations suggérés (Section 4). Des expérimentations démontrent la qualité obtenue par notre algorithme (Section 5).

2 - Travaux existants :

Plusieurs travaux ont été menés pour pallier à ce problème d'extraction de relations dans des documents textuels. Ces travaux permettent aujourd'hui d'avoir des applications pouvant extraire des entités spatiales et les relations topographiques qui existent entre elles.

2.1 - Détection des entités:

Nous avons analysé certaines applications d'extraction d'entités existants afin de pouvoir choisir celle qui répondrait au mieux à nos attentes.
Ainsi notre analyse s'est focalisé sur 3 applications, en l'occurrence DBpédia Spotlight [1], TextRazor [2] et Alchemy API [3].

Ces applications recherchent des entités dans des documents textuels et retournent cette liste d'objets. Chaque objet est une entité qui contient le nom de l'entité et une URL wikipedia. Par exemple avec la phrase " Paris est une capitale" nous obtenons l'entité Lyon et le lien wikipedia *"https://fr.wikipedia.org/wiki/Paris"*.
Dans la suite de ce rapport nous expliquerons nos critères d'évaluation de ces applications et le choix effectué après ces évaluations.

2.2 - Outils d'extraction de relations:

De nombreux outils permettent d'extraire les relations entre des entités présentes dans un document textuel. Leurs performances varient selon les caractéristiques sur lesquelles ils se focalisent (relation binaire, relation souvent exprimée sous forme verbale ...).
Le travail du département "Turing Center" de l'université de Washington [4] propose une comparaison entre deux outils récents ReVerb [5] et R2A2 [6]. Le premier se focalise sur les relations binaires sous forme verbale qui respectent des contraintes syntaxiques et lexicales (e.g., *L'architecture du musée de Valence est similaire à celle du musée de Confluence*), tandis que le deuxième introduit l'analyse linguistique des structures des phrases afin de couvrir le maximum de relations.
Bien que les outils d'extraction de relations aient évolué, cette étude nous montre qu'ils ont encore des limites considérables. Ils ne couvrent que les relations qui se situe entre les entités, dès que la relation se positionne avant la première entité ou après la deuxième entité dans une phrase ReVerb [5] la néglige. De plus ils sont incapables de traiter les relations qui lient plusieurs entités.

3 - Notre approche d'extraction de relation pour le spatial :

Cette partie illustre les différentes méthodes d'extraction d'entités et des relations qui lient ces entités.
La Figure ci-dessous détaille le processus général de notre approche. En entrée, notre approche SMExtract utilise des documents textuels. Ceux-ci sont découpés en phrases par le paragrahe Splitter, chaque phrase est traitée par l'outil de détection d'entités afin d'en extraire les deux entités spatiales à condition qu'ils existent dans la base de données des ressources Wikipedia [7]. Vient ensuite l'étiquetage grammatical de la séquence de mots qui se trouve entre ces deux entités grâce à un outil de POS Tagging [8], ainsi que la détection et la recherche du type de relation dans notre base de données. Et enfin l'intervention de l'expert qui valide les types de relations extraites avant leurs stockage dans la base de données.

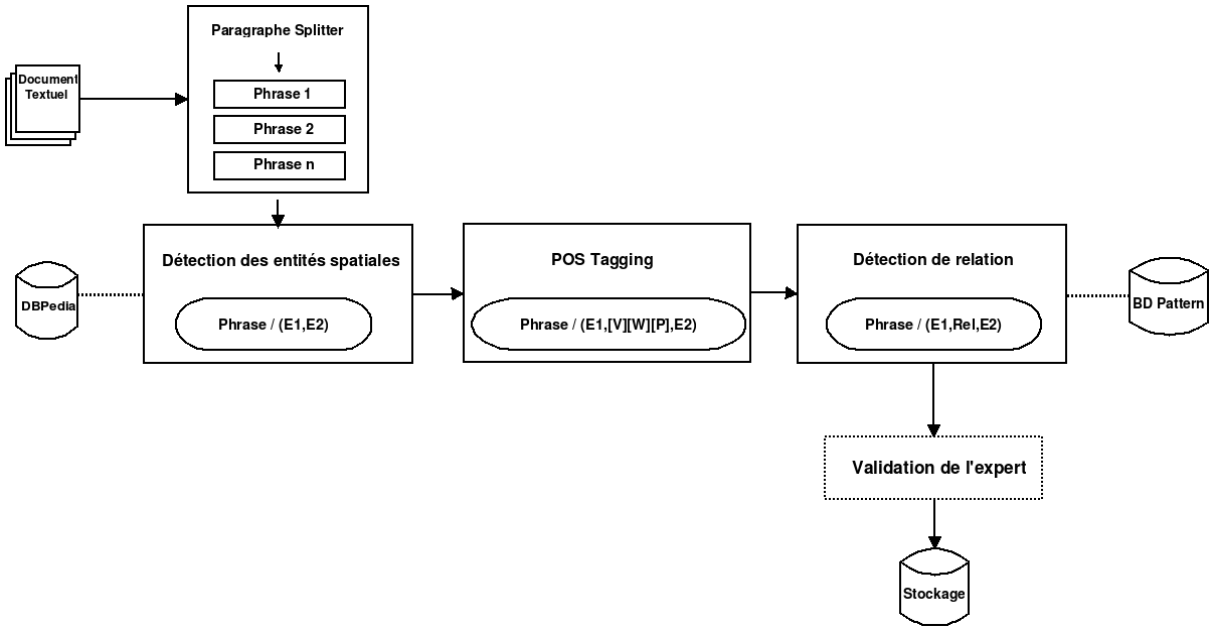

Figure 1 - Fonctionnement de SMExtract

3.1 - Liste des types de relation entre entités spatiales:

Nous avons ci-dessous une liste de quelques types de relations ainsi que des exemples des phrases dans lesquelles ils apparaissent.

Relations topographiques :

Localisation :

- « Le centre commercial La Part-Dieu *se situe* à Lyon ».
- « La ville de Lille *se trouve* dans le nord de la France ».

Inclusion :

- « La France fait *partie* de l'espace Schengen ».
- « Le lac est *dans* la forêt ».

Relations non topographiques :

Modèle :

- « L'architecture du musée de Valence est *similaire* à celle du musée de Confluence ».
- «L'amphithéâtre romain a servi de *modèle* pour les stades actuels».

Esthétique :

- « Le centre commercial Confluence est plus *beau* que celui de la Part-Dieu ».

Administrative :

- « La ville de Bruxelles est la *capitale* de la Belgique ».

- « La ville de Lyon est *jumelée* avec Milan ».

3.2 - Détection des entités :

L'objectif de ce TER n'est pas de détecter les entités spatiales. Aussi, nous avons utilisé un outil existant pour découvrir ces entités spatiales.
Après l'intégration de l'outil nous récupérons les résultats générés et vérifions si les entités retournées sont des ressources Wikipedia [7], et nous vérifions aussi s'il n'y a pas de redondance.
Ensuite, nous formons des paires d'entités. Pour cela, nous parcourons la liste en prenant l'élément i+1 nous vérifions s'il est différent de l'élément i, dans ce cas nous créons un objet composé de ces deux éléments. Cette vérification évite les redondances que nous retourne l'outil d'extraction souvent.
Enfin, nous transmettons le résultat à l'algorithme d'extraction de relation qui s'en servira.

3.3 - Algorithme d'extraction de relations :

Notre algorithme d'extraction de relations prend en paramètre la paire d'entités préalablement détectées dans une phrase, chaque paire est représentée par deux arguments (argument1, argument2) qui représentent les mentions d'entités dans le texte.
Le résultat est obtenu sous forme d'une extraction de relation représentée par le triplet (argument1, relation, argument2).
L'algorithme implémenté dans SMExtract se focalise principalement sur les relations binaires sous forme verbale. Ainsi la relation extraite doit respecter des contraintes syntaxiques que nous définissons sous forme d'étiquetage grammatical (POS tagging). La relation peut être *une phrase verbale simple*, *une phrase verbale suivie par une préposition*, *un article*…, *ou une phrase verbale suivie d'une phrase nominale simple qui se termine par (une préposition, un article …)*.

V | VP | VW* P

V = verbe
W = (nom | adj | adv | pron | det)
P = (prép | art)

Figure 2 - POS Tagging pattern

Pour se faire SMExtract déroule un processus qui se résume en trois étapes :
Premièrement il extrait la séquence de mots qui marque la fin de l'argument i et le début de l'argument i+1. Cette séquence de mots doit absolument respecter les contraintes syntaxiques citées ci-dessus.

Ensuite il compare cette séquence aux connaissances (patterns) présentes dans notre base de données, la comparaison se fait en calculant la distance de Levenshtein [9]. À la fin de la deuxième étape nous obtenons un tableau qui associe chaque pattern à un pourcentage de ressemblance avec la séquence extraite. Lors de la troisième étape nous trions le tableau en ordre décroissant ensuite nous sélectionnons les types de relations qui correspondent aux cinq patterns avec le pourcentage le plus élevé.

4 - Développement de l'application SMExtract :

SMExtract est une application web développée en Java EE. La gestion de sa production est réalisée grâce à l'outil Apache Maven. SMExtract est liée à la base de données relationnelle H2, les persistances sont effectuées avec la bibliothèque JPA . Elle intègre l'outil NLP TextRazor [2].
Tout au long du développement de l'application les différents tests ont été effectués avec Junit Test. Pour l'interface Homme/Machine nous avons choisi d'utiliser des pages JSP, la bibliothèque Bootstrap pour le CSS et AJAX pour l'interaction avec l'utilisateur.

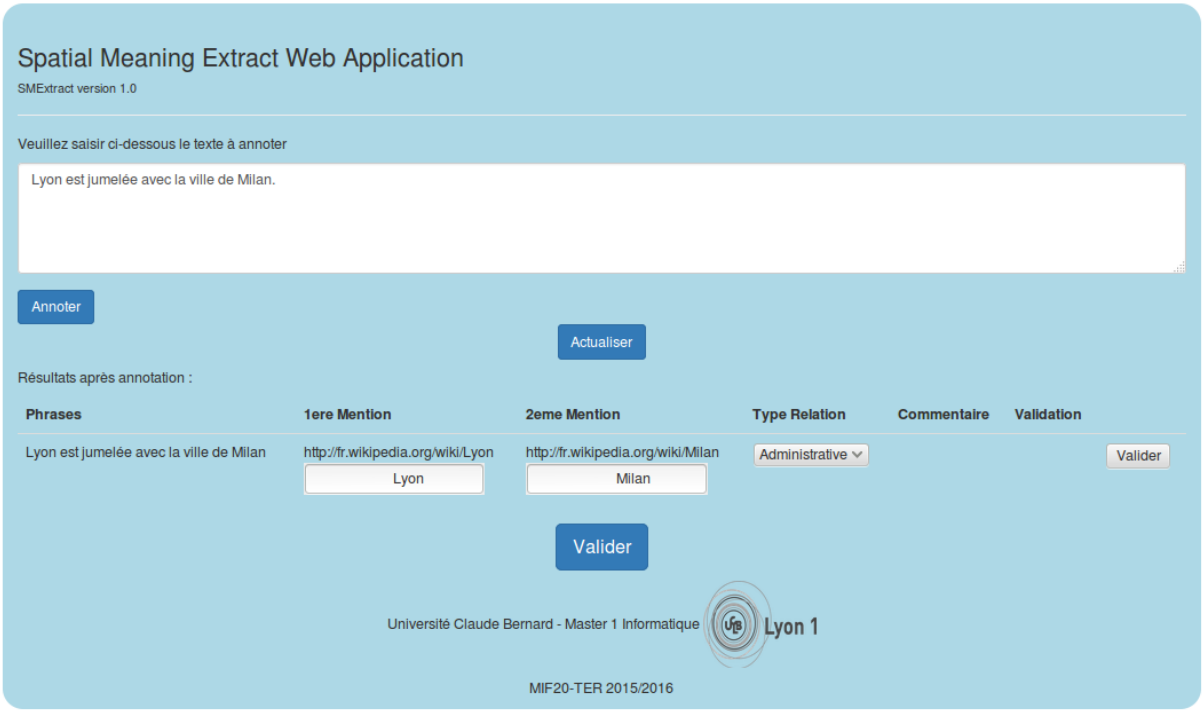

Figure 3 – capture d'écran de l'application SMExtract

5 - Validation expérimentale :

5.1 - Protocole d'évaluation :

Notre but est à partir d'un document textuel donné de rechercher et trouver des informations pertinentes et uniquement celles-là. C'est principalement sur ce critère que nous nous basons pour évaluer la capacité des différents outils d'extraction.
Nous avons sélectionné deux documents textuels, le premier que nous avons soigneusement saisis contient des entités spatiales simples à repérer à l'inverse du deuxième, un texte Wikipedia plus complexe. Les deux documents textuels constituent notre corpus sur lequel nous allons effectuer les évaluations. Nous avons ensuite défini une classe d'entités spatiales à partir de ce corpus.
Ainsi lorsque l'outil retourne une réponse par rapport au corpus et à la classe, deux choix s'offrent à lui :

- L'entité appartient selon lui à la classe.
- L'entité n'appartient pas selon lui à la classe.

En face de ces deux possibilités de réponses de l'outil, nous avons les deux cas où :

- L'entité appartient à la classe.
- L'entité n'appartient pas à la classe.

Pour évaluer le jeu de données, nous utilisons deux mesures : la mesure de Précision et de Rappel . La précision est définie comme étant le rapport entre les entités extraites correctes (VP) sur l'ensemble des entités découvertes (VP+ FP). Le rappel est quant à lui défini par le rapport entre les entités extraites correctes (VP) sur le nombre total d'entités qu'il fallait découvrir (VP + FN).

Précision = VP / (VP + FP) **Rappel = VP / (VP + FN)**

Figure 4 - Formules Précision/Rappel

5.2 - Choix d'un outil de détection d'entités :

L'analyse des trois outils d'extraction d'entités selon nos critères d'évaluation nous a permis de choisir le TextRazor [2] car il répondait au mieux à nos attentes. Vous trouverez dans les tableaux ci-dessous les résultats obtenus.

Outil	Précision	Rappel	VP	FP	FN
TextRazor	58%	22%	37	26	125
DBpedia Spotlight	32%	59%	49	101	33
Alchemy API	37%	50%	37	62	36

Figure 5 - Tableau de rappel et précision des outils

Outil	1175 caractères	2500 caractères	5000 caractères
TextRazor	2s15	2s66	3s03
DBpedia Spotlight	1s21	1s91	2s36
Alchemy API	2s03	2s70	3s16

Figure 6 - tableaux de performance en fonction du temps d'exécution

5.3 - Évaluation de la détection de relations :

Cette évaluation a pour objectif de vérifier la pertinence de l'algorithme d'extraction de relations à l'aide d'un corpus textuel et d'une classe de types de relations que nous avons définies. Nous voulons vérifier si l'algorithme détecte et retourne le type de relation correct en premier, c'est à dire avec le score de pertinence le plus élevé. Pour cela, nous avons utilisé la mesure de précision décrites précédemment. Tout d'abord, le calcul des statistiques est effectué sur le premier type de relation détecté pour chacune des relations (Top-1). Puis il est effectué sur les deux premiers types de relations détectés (Top-2) et ainsi de suite jusqu'à (Top-5).

Sur la Figure ci-dessous, nous constatons que lorsque l'utilisateur vérifie uniquement le premier type de relation détecté, celui-ci est précis à près de 80%. Tandis qu'en arrivant au Top-5, on remarque que 100% des types de relations détectés sont corrects.

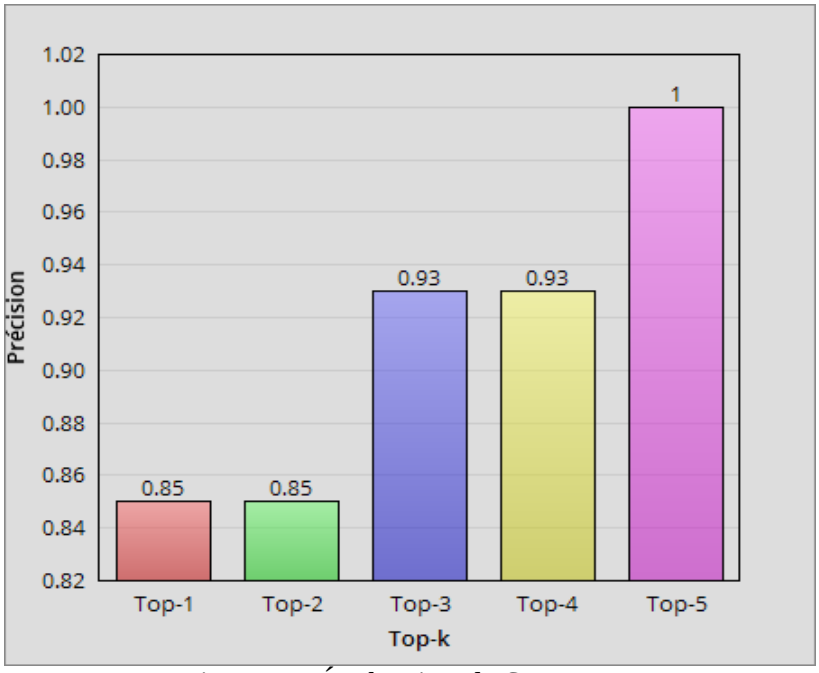

Figure 7 - Évaluation de SMExtract

6 - Conclusion :

Le but de ce projet était de construire une base de données d'apprentissage pour des relations spatiales. Pour cela, nous devions mettre en place une application permettant d'extraire des entités spatiales dans des documents textuels ainsi que les relations non topographiques qui existent entre ces entités.
Pour mener à bien cette mission, nous avons dans un premier temps fait une étude sur un ensemble d'outils d'extraction d'entités telque DBpédia Spotlight [1], TextRazor [2], Alchemy API [3]. L'une des difficultés majeures rencontrées était le choix d'un outil d'extraction d'entités car nous n'avions pas encore pris connaissance de ces outils avant le projet. À la suite d'une série d'expérimentations, nous avons alors choisi le TextRazor [2] et l'avons intégré à notre application nommée SMExtract.

Ensuite, nous avons mis en place un notre algorithme d'extraction de relations non topographiques entre les entités détectées, la difficulté rencontrée à ce niveau était de rendre cet algorithme générique afin qu'il couvre un maximum de relations.
Cet algorithme se base sur un ensemble de patterns flexibles définis dans la base de données. En calculant la distance de Levenshtein entre les relations présentes dans le texte et les patterns, nous parvenons à en déduire le type de relation correspondant.

Ce TER nous a permis de nous familiariser avec le monde de la recherche.
Grâce à ce projet nous avons acquis un minimum d'expérience sur la recherche d'informations dans des documents textuels.

7 - Références :

[1] Lien vers l'outil DBpedia Spotlight Cliquez ici
[2] Lien vers l'outil TextRazor Cliquez ici
[3] Lien vers l'outil Alchemy API Cliquez ici
[4] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open information extraction The second generation. In IJCAI, volume 11, pages 3–10, 2011. Lien vers IJCAI11-OIE.pdf
[5] lien vers l'outil ReVerb Cliquez ici
[6] Lien vers l'outil R2A2 Cliquez ici
[7] Lien vers wikipedia Cliquez ici
[8] Liens vers la définition de POS Tagging Cliquez ici
[9] Liens vers la définition de Levenshtein Cliquez ici