ADEL : une méthode adaptative de désambiguïsation d'entités nommées

Julien Plu[1], Raphaël Troncy[1], Giuseppe Rizzo[2]

[1] EURECOM, Sophia-Antipolis, France
julien.plu@eurecom.fr, raphael.troncy@eurecom.fr

[2] ISMB, Turin, Italie giuseppe.rizzo@ismb.it

Résumé : Nous identifions quatre critères principaux pouvant causer de sérieuses difficultés lorsqu'il s'agit de développer un système de désambiguïsation d'entités nommées : i) la nature du document à analyser (tweet, sous-titre d'une vidéo, article de presse) ; ii) le nombre de types possibles qui sont utilisés pour catégoriser une mention (Personne, Lieu, Date, Rôle) ; iii) la base de connaissances utilisée pour désambiguïser les mentions extraites (DBpedia, Wikidata, Musicbrainz) ; iv) la langue utilisée dans le document. Dans cet article, nous présentons ADEL, une approche innovante basée sur une architecture hybride pour un système de désambiguïsation d'entités nommées combinant des méthodes du domaine du Traitement Automatique du Langage Naturel (TALN), de la recherche d'information et du Web Sémantique. En particulier, nous proposons une approche modulaire afin d'être aussi indépendants que possible du texte analysé et de la base de connaissance utilisée. Notre évaluation montre que ce système obtient des résultats comparables ou meilleurs que l'état de l'art sur cinq jeux de données : OKE2015, OKE2016, NEEL2014, NEEL2015 et NEEL2016.

Mots-clés : Désambiguïsation d'entités nommées, reconnaissance d'entités nommées, extraction d'information

1 Introduction

Il y a une augmentation exponentielle de contenu textuel disponible sur le Web, notamment produit par les internautes via une large diversité de plate-forme de publication et qui a besoin d'être traité automatiquement. Concernant ce contenu textuel, nous avons identifié quatre défis principaux pour la communauté du TALN : gérer différents types de textes (billets sur les réseaux sociaux, requêtes d'un moteur de recherche, sous-titres d'une vidéo, articles de journaux) écrits dans des langues différentes ; détecter des entités nommées propres à des domaines variés pouvant être classifiées et désambiguïsées avec des classes et des graphes de connaissances spécifiques. Ces défis affectent la stratégie à utiliser pour comprendre le texte et pour extraire et désambiguïser des unités d'informations pertinentes. Différentes bases de connaissances ont été utilisées pour une telle tâche : DBpedia, Freebase, Wikidata [1] pour n'en nommer que quelques-unes. Ces bases de connaissances sont connues pour être large en termes de couverture, alors que des bases de connaissances verticales existent aussi dans des domaines plus spécifiques, par exemple, Geonames, Musicbrainz ou LinkedMDB [2]. Ces quatre défis sont bien représentés dans différents jeux de donnée de références (van Erp *et al.*, 2016), où chaque jeu de données a ses propres règles d'annotation : différent contenu textuel (articles de journaux pour AIDA, tweets pour NEEL), différentes définitions des types des entités nommées (Location pour NEEL, Place pour OKE), différentes base de connaissances (Freebase pour TAC-KBP, DBpedia pour OKE), différentes langues (anglais pour MEANTIME, français pour ETAPE (Gravier *et al.*, 2012)).

1. http://dbpedia.org, https://www.freebase.com, https://www.wikidata.org/
2. http://www.geonames.org, https://musicbrainz.org, http://www.linkedmdb.org

La désambiguïsation et la reconnaissance d'entités nommées sont deux sous-tâches de l'extraction d'information. Dans cet article, nous appelons une *mention* - une expression extraite d'un texte, une *entité nommée* - une ressource décrite dans une base de connaissances, une *entité nommée candidate* - une possible entité nommée pour une mention, et une *annotation* comme le couple *(mention, entité nommée)* définissant le lien établi d'une mention dans le texte à la base de connaissance référente. Le principe de la reconnaissance d'entités nommées est d'extraire des mentions et de leur donner un type conformément à un ensemble de classes prédéfinies (par exemple Personne, Lieu ou Organisation). Le principe de désambiguïsation d'entités nommées est d'annoter les mentions extraites à partir du texte avec leur entité correspondante décrite dans une base de connaissances. Chaque entité nommée se voit associer une liste de candidats. Toutes les entrées dans la base de connaissance représentent une entité du monde réel de manière unique avec un identifiant spécifique. Les entités nommées qui n'apparaissent pas dans la base de connaissance sont généralement appelées *entités nommées émergentes* ou *NIL*. Lorsque l'on analyse du contenu textuel, l'ambiguïté et la synonymie sont des problèmes fondamentaux dont il faut s'occuper. Une entité nommée peut avoir plus d'une mention (synonymie) et une mention peut représenter plus d'une entité nommée (ambiguïté). Dans cet article, nous proposons une approche de reconnaissance et désambiguïsation d'entités nommées permettant de répondre à ces quatre défis. Nous détaillons notre approche et le système ADEL dans la Section 2 et nous décrivons plusieurs évaluations dans la Section 3 avant de conclure (Section 4).

2 ADEL : une architecture modulaire

ADEL est composé de multiples modules qui peuvent être activés ou désactivés. Ces modules sont regroupés dans trois composants : *Entity Extraction*, *Index* et *Entity Linking* (Figure 2). ADEL est implémenté en Java et est publiquement disponible comme une API REST [3]. ADEL s'attaque aux défis mentionnés dans l'introduction en adaptant ses fonctionnalités au texte, à la langue, au type des entités nommées et à la base de connaissance.

2.1 Entity Extraction

Le premier module du composant *Entity Extraction* est le *Extractors Module*. Il extrait les mentions à partir du texte qui sont susceptibles d'être sélectionnées comme entités nommées. Après avoir identifié les mentions, nous résolvons leur potentielle superposition en utilisant le module *Overlap Resolution Module*.

Nous utilisons de multiples extracteurs : dictionnaire, étiquetage morphosyntaxique (POS), reconnaissance d'entités nommées (NER), co-référence, nombre et date. Chacun de ces extracteurs fonctionne en parallèle et peut être basé sur des systèmes de TALN extérieurs comme Stanford CoreNLP, GATE ou NLTK. Nous avons développé une API Web générique pouvant être implémentée pour chaque système de TALN afin de le rendre utilisable par ADEL. Comme exemple, la version de cette API pour Stanford CoreNLP est disponible sur Github [4]. Nous fournissons aussi une documentation de cette API [5] afin de permettre à tout le monde de développer

3. `http://adel.eurecom.fr/api/`
4. `https://github.com/jplu/stanfordNLPRESTAPI`
5. `https://github.com/jplu/stanfordNLPRESTAPI/wiki/API-documentation`

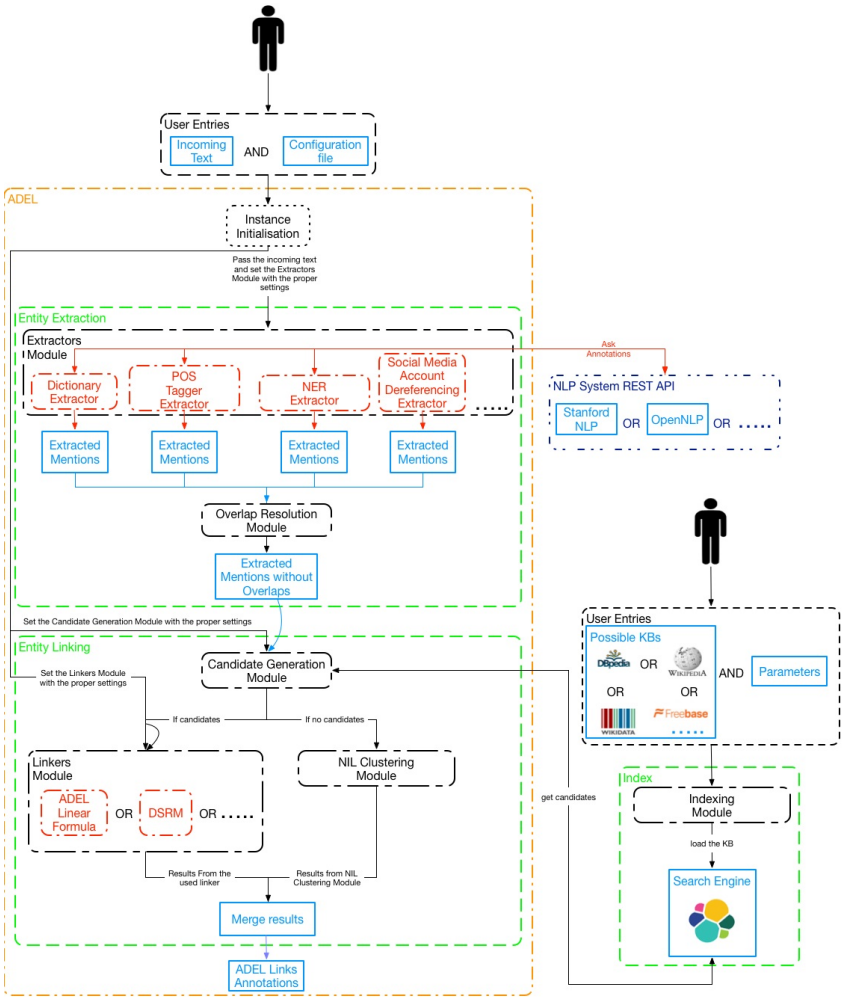

FIGURE 1 – Architecture du système ADEL

son propre extracteur à intégrer dans ADEL. Il n'y a pas de limite au nombre d'extracteurs et de modèles pouvant être utilisé dans ADEL. L'API Web de chaque extracteur est utilisé comme composant extérieur, et a besoin d'être instancié en amont. Il est possible d'utiliser une combinaison de modèles permettant de faire usage de différents modèles CRF. Soit f une fonction qui prend un modèle m et un texte txt et produit un ensemble de tuples $(mention, etiquette)$: $f_m(txt) = \{(c_1, l_1), \ldots, (c_k, l_k)\}$. La combinaison des modèles revient à faire l'union des résultats obtenus pour chaque modèle considéré : $\bigcup_{m \in M} f_m(txt)$. L'ordre dans lequel les modèles sont appliqués est important. Par conséquent, si une mention est incorrectement annotée par le premier modèle, le second modèle ne pourra pas le corriger même si ce second modèle aurait pu annoter correctement cette mention. Cet algorithme dans Stanford NER est appelé *NER Classifier Combiner*.

Un extracteur peut extraire des mentions qui ont une superposition partielle ou totale avec les autres mentions extraites par d'autres extracteurs. Afin de résoudre cette ambiguïté, nous avons implémenté un module de résolution de superposition qui prend la sortie de chaque extracteur et donne une sortie sans superpositions. La logique de ce module est la suivante : étant donné deux

mentions qui se superposent, par exemple `Unis d'Amérique` venant de l'extracteur NER et `Etats-Unis` venant de l'extracteur POS, nous prenons l'union des deux mentions. Nous obtenons la mention `Etats-Unis d'Amérique` et le type fourni par l'extracteur NER est sélectionné. Nous avons aussi implémenté d'autres heuristiques afin de résoudre les superpositions, mais le choix de la bonne heuristique à utiliser est laissé à une configuration manuelle.

2.2 Index

Des bases de connaissances différentes sont utilisées en fonction des *challenges* de référence (par exemple, Wikipedia et ensuite Freebase comme base de connaissances pour TAC-KBP). Il y a donc un besoin d'avoir une façon générique et performante d'indexer n'importe quelle base de connaissances. Un index peut être vu comme un tableau à deux dimensions où chaque ligne est une entité dans l'index et chaque colonne est une propriété qui décrit par un littéral cette entité. Ainsi, indexer entièrement la version anglaise de DBpedia donne 281 colonnes. Une fois que nous avons cet index, nous pouvons chercher une mention dans cet index et récupérer les entités nommées candidates. Chercher, par défaut, sur toutes les colonnes (ou les propriétés utilisées dans la base de connaissances), impact de manière négative la performance de l'index en termes de temps de calcul. Afin d'optimiser l'index, nous avons développé une heuristique qui vise à maximiser la couverture de l'index tout en minimisant le nombre de colonnes (ou propriétés) sur lesquelles effectuer une recherche. Pour la version 2015-10 de DBpedia, il y a exactement 281 propriétés ayant des valeurs littérales, alors que notre méthode d'optimisation réduit cette liste à 8 propriétés : `dbo:wikiPageWikiLinkText`, `dbo:wikiPageRedirects`, `dbo:demonym`, `dbo:wikiPageDisambiguates`, `dbo:birthName`, `dbo:alias`, `dbo:abstract` et `rdfs:label`. Cette optimisation réduit drastiquement le temps de la requête en passant d'environ 4 secondes à moins d'une seconde. Le code source de cette optimisation est aussi disponible [6]. L'index est construit en utilisant *Elasticsearch* comme moteur de recherche. L'indexation d'une base de connaissances suit un processus à deux étapes : *i)* extraire le contenu de la base de connaissances et créé l'index Elasticsearch avec ces données ; *ii)* lancer la méthode d'optimisation afin de récupérer la liste des colonnes qui seront utilisées pour interroger l'index.

2.3 Entity Linking

Le composant *Entity Linking* commence avec le *Candidate Generation Module* qui interroge l'index et génère une liste d'entités nommées candidates pour chaque mention extraite. Si l'index retourne une liste d'entités nommées candidates, alors le *Linkers Module* est invoqué ; alternativement, si une liste vide est retournée, alors le *NIL Clustering Module* est invoqué.

Le *NIL Clustering Module* propose de regrouper les entités nommées *NIL* (entités nommées émergentes) qui peuvent identifier la même chose dans le monde réel. Le rôle de ce module est d'attacher la même valeur *NIL* dans un même document. Par exemple, si deux mentions extraites partagent la même entité nommée émergente, ces mentions seront annotées avec la même valeur *NIL_1*.

Le *Linkers Module* permet d'offrir plusieurs méthodes. Jusque là, nous avons intégré une seule méthode appelée *ADEL Formula*, mais nous sommes actuellement en train de dévelop-

6. `https://gist.github.com/jplu/a16103f655115728cc9dcff1a3a57682`

per d'autres méthodes qui seront aussi intégrées dans ADEL. Ces futures méthodes seront basées sur D2V Le & Mikolov (2014), TF-IDF avec une distance cosinus, ainsi qu'adapter DSSM Huang *et al.* (2013) pour la désambiguïsation d'entités nommées. Nous avons aussi développé une fonctionnalité dans ADEL permettant l'intégration d'une méthode de désambiguïsation extérieure en respectant l'implémentation d'une interface Java. La méthode *ADEL Formula* se résume à l'Equation 1.

$$r(l) = (a \cdot L(m, label) + b \cdot max(L(m, R)) + c \cdot max(L(m, D))) \cdot PR(l) \quad (1)$$

La fonction $r(l)$ utilise la distance de Levenshtein L entre la mention m et le label, la distance maximum entre la mention m et chaque élément (label) dans l'ensemble de pages de redirection de Wikipedia R et la distance maximum entre la mention m et chaque élément (label) dans l'ensemble de pages de désambiguïsation de Wikipedia D, pondéré par le PageRank PR, pour chaque entité nommée candidate l. Les poids a, b et c sont une combinaison convexe devant satisfaire : $a + b + c = 1$ et $a > b > c > 0$. Nous prenons l'hypothèse que la distance entre une mention et un label est plus importante que la distance avec la page de redirection et qui est elle-même plus importante que la distance avec la page de désambiguïsation.

3 Evaluation

Notre approche a été évaluée avec les jeux de données tests de plusieurs *challenges* : NEEL-2014 (Basave *et al.*, 2014), NEEL2015 (Rizzo *et al.*, 2015) et NEEL2016 (Rizzo *et al.*, 2016), ainsi que OKE2015 (Nuzzolese *et al.*, 2015) et OKE2016 (Nuzzolese *et al.*, 2016). Nous présentons une évaluation de chaque composant pour chacun de ces jeux de données en utilisant le *scorer* TAC-KBP (Hachey *et al.*, 2014) : le Tableau 1 montre les performances d'ADEL pour chacun de ses composants et le Tableau 2 montre les meilleurs scores parmi tous les participants de ces *challenges*.

Niveau	**OKE2015**	**OKE2016**	**NEEL2014**	**NEEL2015**	**NEEL2016**
strong_mention_match	71.2	78.2	70.8	71.6	85.5
strong_typed_mention_match	59.8	72.0	-	52.8	61
strong_link_match	48	49.1	46.3	47.9	53.8

TABLE 1 – Résultats d'ADEL avec le *scorer* NELEVAL en F-mesure

Niveau	**OKE2015**	**OKE2016**	**NEEL2014**	**NEEL2015**	**NEEL2016**
strong_mention_match	**71.2**	**78.2**	-	-	**85.5**
strong_typed_mention_match	**59.8**	**72.0**	-	80.7	**61**
strong_link_match	**48**	60.08	70.6	76.2	**53.8**

TABLE 2 – Meilleur score parmi tous les participants des *challenges* en F-mesure. Les scores en gras sont ceux d'ADEL.

Les résultats pour les *challenges* OKE montrent que nous sommes meilleurs dans tous les cas (sauf un). Les résultats pour les *challenges* NEEL montrent que notre approche est robuste pour l'extraction et la classification des entités nommées voir meilleurs que les autres participants pour l'édition 2016. D'une manière générale, les résultats montrent une chute importante au moment de la désambiguïsation, principalement due à une approche non supervisée.

4 Conclusion

Nous avons démontré qu'ADEL permet de s'adapter aux quatre défis suivants : la **langue** (en changeant les modèles et les dictionnaires utilisés par le(s) système de TALN) ; la nature du **texte** (tweets, sous-titre de vidéos, articles de presse) ; les **classes des entités nommées** (des types conventionnels comme Personne, Lieu ou Organisation à des types plus précis comme Sportif ou Chanteur) ; la **base de connaissances** (DBpedia ou MusicBrainz). Comme futur travail, nous prévoyons d'intégrer trois nouvelles méthodes de désambiguïsation : D2V, TF-IDF avec la similarité de cosinus et DSSM. Nous prévoyons aussi d'améliorer le processus d'extraction pour *1)* améliorer la prise en charge des *hashtags* en intégrant une méthode de segmentation performante pour améliorer la génération de candidats pour les tweets ; *2)* implémenter une association entre les types provenant de plusieurs modèles (Rizzo *et al.*, 2014) afin d'améliorer le raffinement des types.

Références

BASAVE A. E. C., RIZZO G., VARGA A., ROWE M., STANKOVIC M. & DADZIE A. (2014). Making Sense of Microposts (#Microposts2014) Named Entity Extraction & Linking Challenge. In *4th Workshop on Making Sense of Microposts*, Seoul, Korea.

GRAVIER G., ADDA G., PAULSSON N., CARRÉ M., GIRAUDEL A. & GALIBERT O. (2012). The ETAPE corpus for the evaluation of speech-based TV content processing in the French language. In *8th International Conference on Language Resources and Evaluation (LREC)*.

HACHEY B., NOTHMAN J. & RADFORD W. (2014). Cheap and easy entity evaluation. In *52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

HUANG P.-S., HE X., GAO J., DENG L., ACERO A. & HECK L. (2013). Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *22nd ACM International Conference on Information & Knowledge Management (CIKM)*.

LE Q. V. & MIKOLOV T. (2014). Distributed Representations of Sentences and Documents. In *31st International Conference on Machine Learning (ICML)*.

NUZZOLESE A., GENTILE A., PRESUTTI V., GANGEMI A., GARIGLIOTTI D. & NAVIGLI R. (2015). The 1st Open Knowledge Extraction Challenge. In *12th European Semantic Web Conference (ESWC)*.

NUZZOLESE A., GENTILE A., PRESUTTI V., GANGEMI A., MEUSEL R. & PAULHEIM H. (2016). The 2nd Open Knowledge Extraction Challenge. In *13th European Semantic Web Conference (ESWC)*.

RIZZO G., BASAVE A. E. C., PEREIRA B. & VARGA A. (2015). Making Sense of Microposts (#Microposts2015) Named Entity rEcognition and Linking (NEEL) Challenge. In *5th Workshop on Making Sense of Microposts*, Florence, Italy.

RIZZO G., VAN ERP M., PLU J. & TRONCY R. (2016). NEEL 2016 : Named Entity rEcognition & Linking challenge report. In *6th International Workshop on Making Sense of Microposts*.

RIZZO G., VAN ERP M. & TRONCY R. (2014). Inductive Entity Typing Alignment. In *2nd International Workshop on Linked Data for Information Extraction (LD4IE)*.

VAN ERP M., MENDES P. N., PAULHEIM H., ILIEVSKI F., PLU J., RIZZO G. & WAITELONIS J. (2016). Evaluating Entity Linking : An Analysis of Current Benchmark Datasets and a Roadmap for Doing a Better Job. In *10th International Conference on Language Resources and Evaluation (LREC)*.