UNIVERSITÉ DE MONTRÉAL

AMÉLIORATION DES PERFORMANCES DES ANNOTATEURS SÉMANTIQUES

MOHAMED CHABCHOUB
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

AMÉLIORATION DES PERFORMANCES DES ANNOTATEURS SÉMANTIQUES

présenté par : CHABCHOUB Mohamed
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. FERNANDEZ José M., Ph. D., président
M. GAGNON Michel, Ph. D., membre et directeur de recherche
Mme ZOUAQ Amal, Ph. D., membre et codirectrice de recherche
M. LAPALME Guy, Ph. D., membre

DÉDICACE

Je dédie ce travail,
à mon père Chokri et ma mère Amel
et à mes deux frères Mahdi et Karim

REMERCIEMENTS

Je tiens à remercier en premier lieu mon directeur de recherche Michel Gagnon, et ma codirectrice de recherche Amal Zouaq, ils m'ont été d'une aide précieuse durant ces deux dernières années. Je les remercie aussi pour leur patience, leur disponibilité ainsi que leurs conseils judicieux qui m'ont permis de rédiger ce mémoire. Je tiens aussi à remercier mes parents qui ont toujours cru en moi et qui m'ont soutenu tout au long de mes études.

RÉSUMÉ

Les annotateurs sémantiques jouent un rôle important dans la transition du Web actuel au Web sémantique. Ils s'occupent d'extraire des informations structurées à partir de textes bruts, permettant ainsi de pointer vers des bases des connaissances telles que DBpedia, YAGO ou Babelnet. De nombreuses compétitions sont organisées chaque année pour promouvoir les travaux de recherche de ce domaine. Nous présentons dans ce mémoire notre participation à la compétition *Open Knowledge Extraction* que nous avons remportée à la conférence European Semantic Web Conference 2016. Dans le cadre de cette compétition, nous avons implémenté une approche générique que nous avons testée sur quatre annotateurs sémantiques.

Nous nous concentrons dans ce mémoire à décrire un annotateur sémantique en particulier, DBpedia Spotlight. Nous exposons les différentes limites que présente cet annotateur ainsi que les approches que nous avons développées pour y remédier. Nous avons noté une augmentation d'une moyenne de 20% des performances actuelles de DBpedia Spotlight en testant sur différents corpus. Ces derniers proviennent principalement de journaux internationaux, "Reuters News Stories", "MSNBC" et le "New York Times".

ABSTRACT

Semantic annotators play an important role in the transition from the current Web to the Semantic Web. They take care of extracting structured information from raw texts and thus make it possible to point to knowledge bases such as DBpedia, YAGO or Babelnet. Many competitions are organized every year to promote research works in this field. We present in this thesis our system which was the winner of the Open Knowledge Extraction challenge at the European Semantic Web Conference 2016. For this competition, we implemented a generic approach tested with four semantic annotators.

We particularly focus in this thesis on one semantic annotator, DBpedia Spotlight. We present its different limitations along with the approaches that we have developed to remedy them. We noted an improvement of an average of 20% of the current performance of DBpedia Spotlight on different corpora that mainly come from international newspapers, "Reuters News Stories", "MSNBC" and the "New York Times".

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE DES MATIÈRES vii

LISTE DES TABLEAUX x

LISTE DES FIGURES xi

LISTE DES SIGLES ET ABRÉVIATIONS xii

CHAPITRE 1 INTRODUCTION 1

CHAPITRE 2 REVUE DE LITTÉRATURE 3
2.1 Définitions générales 3
2.2 Le Web des données 4
2.3 Architecture des annotateurs sémantiques 4
2.3.1 La détection des mentions 5
2.3.2 Génération des candidats 6
2.3.3 La désambiguïsation 7
2.4 Mise en contexte 13

CHAPITRE 3 MÉTHODOLOGIE 15
3.1 Amélioration de DBpedia Spotlight avec les services FICLONE 15
3.1.1 Le choix de DBpedia Spotlight 15
3.1.2 Les limites de DBpedia Spotlight 16
3.1.3 Amélioration de Spotlight pour la désambiguïsation d'entités nommées (FICLONE NED) 19
3.1.4 Amélioration de Spotlight comme annotateur sémantique (FICLONE SA) 22
3.2 Expérimentation dans le cadre du OKE challenge 2016 (WESTLAB) 23

3.3 Protocole d'évaluation 25

CHAPITRE 4 ARTICLE 1 : FICLONE : IMPROVING DBPEDIA SPOTLIGHT USING NAMED ENTITY RECOGNITION AND COLLECTIVE DISAMBIGUATION . 27
4.1 Introduction 27
4.2 Related work 29
4.3 FICLONE 30
4.3.1 Named entity disambiguation using FICLONE NED 31
4.3.2 Semantic annotation using FICLONE SA 37
4.4 Evaluation methodology 38
4.4.1 Experimental setup 38
4.4.2 Selection of metric for coherence score 40
4.4.3 Performance of state-of-the-art annotators 41
4.4.4 Evaluation of FICLONE NED 42
4.4.5 Evaluation of FICLONE SA 44
4.5 Analysis of results 46
4.5.1 Analysis of spotting results 47
4.5.2 Analysing the disambiguation step 48
4.6 Conclusion 50
4.7 Aknowledgement 51

CHAPITRE 5 ARTICLE 2 : COLLECTIVE DISAMBIGUATION AND SEMANTIC ANNOTATION FOR ENTITY LINKING AND TYPING 52
5.1 Introduction 52
5.2 Related work 53
5.2.1 Named entity recognition 53
5.2.2 Semantic annotation 54
5.2.3 Entity typing 55
5.3 System implementation 55
5.3.1 System Architecture 55
5.3.2 Entity Spotting 56
5.3.3 Entity Linking 58
5.3.4 Entity Typing 58
5.4 Experiments and results 60
5.4.1 Training set description 60
5.4.2 Results on the OKE challenge training dataset 60
5.4.3 Baseline comparisons 61

5.4.4 Result on the OKE Challenge evaluation dataset 62
5.4.5 Analysis of errors 64
5.5 Conclusion 67
5.6 Aknowledgement 67

CHAPITRE 6 DISCUSSION GÉNÉRALE 68

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS 70

RÉFÉRENCES 71

LISTE DES TABLEAUX

Tableau 3.1	Annotation pour le OKE challenge	24
Table 4.1	Statistics on the datasets	40
Table 4.2	Comparison of F-scores obtained with Jaccard and relatedness metrics for disambiguation	41
Table 4.3	Performances of NED and SA systems	41
Table 4.4	Performances of Stanford NER and DBpedia Spotlight for the entity spotting substask	42
Table 4.5	Comparison with other NED systems for spotting (ES)	43
Table 4.6	Comparison with other NED systems for disambiguation (ED)	44
Table 4.7	Comparison with other NED systems for the full process (A2KB)	45
Table 4.8	Comparison with other SAs for the full process (A2KB)	45
Table 4.9	Comparison with other SAs for spotting (ES)	45
Table 4.10	Comparison with other SAs for disambiguation (ED)	45
Table 4.11	Impact on spotting	47
Table 4.12	Entity disambiguation analysis	48
Table 5.1	Statistics on the training set per DUL type	60
Table 5.2	Results of the entity recognition, linking and typing tasks on the training dataset	61
Table 5.3	Comparison with Stanford NER	61
Table 5.4	Comparison with semantic annotators	62
Table 5.5	Statistics on the evaluation set per DUL type	63
Table 5.6	Results of the entity recognition, linking and typing tasks on the evaluation dataset	63
Table 5.7	Comparison with semantic annotators	63
Table 5.8	Results on the evaluation dataset at the OKE challenge 2016, with weak match	64
Table 5.9	Spotting errors, according to their origin.	65
Table 5.10	Disambiguation errors	66
Table 5.11	Typing errors	66

LISTE DES FIGURES

Figure 2.1	Architecture générale des annotateurs sémantiques	5
Figure 3.1	Exemples d'erreurs de reconnaissance des mentions de DBpedia Spotlight 0.5	16
Figure 3.2	Exemples d'erreurs de reconnaissance des mentions de DBpedia Spotlight 0.0	17
Figure 3.3	Exemple d'erreur de génération de candidats DBpedia Spotlight	18
Figure 3.4	Requetes SPARQL	20
Figure 3.5	Description des entités rentrantes et sortantes	21
Figure 4.1	FICLONE NED architecture	32
Figure 4.2	Spotlight Snapshot	33
Figure 4.3	DBpedia Spotlight outputs	34
Figure 4.4	FICLONE SA architecture	38
Figure 4.5	DBpedia Spotlight errors	43
Figure 4.6	FICLONE errors	49
Figure 4.7	The difficulty of Entity Linking	50
Figure 5.1	System architecture	56

LISTE DES SIGLES ET ABRÉVIATIONS

NED	Désambiguïsation des entités nommées *(Named Entity Disambiguation)*
POS	Catégorie morphosyntaxique *(Part-of-speech)*
NER	Reconnaissance des entités nommées *(Named Entity Recognition)*
SA	Annotation Sémantique *(Semantic Annotation)*
OKE	Open Knowledge Extraction
NIF	NLP Interchange Format
RDF	Resource Description Framework
TF	Term Frequency
IDF	Inverse Document Frequency
ICF	Inverse Candidate Frequency

CHAPITRE 1 INTRODUCTION

De nos jours, la quantité de données présentes sur internet ne cesse d'augmenter de manière exponentielle avec des blogues, des journaux, des forums de discussion et d'autres types d'informations en tout genre. Pour l'instant, ces données sont interprétables uniquement par les humains. Ces derniers doivent précisément rechercher l'information désirée car les machines ne peuvent, à ce jour, que les rapprocher de l'information en proposant des suggestions de documents.

En 2001, Tim-Berners Lee a introduit l'idée du Web sémantique. Un Web dans lequel les machines et les humains coopèrent. Comment ? En ajoutant une couche d'information structurée pour qu'elle soit exploitable par des machines. L'un des standards, parmi les plus importants, qui ont été mis en place par l'organisme W3C est le RDF [1]. Celui-ci permet de décrire des ressources Web sous une forme bien structurée, permettant ainsi de rendre l'information plus accessible et compréhensible par les machines. Cependant, le Web actuel contient environ 45 milliards de pages [2], dont la plupart sont non structurées. De plus, la structuration de ces pages selon le format voulu (RDF) requiert l'implémentation de diverses techniques. Parmi ces techniques, nous trouvons l'extraction de triplets en RDF à partir de textes non structurés. Il s'agit en fait d'extraire les entités les plus pertinentes du texte, de les lier à une base de connaissances pour que ces entités soient bien identifiées et d'enfin extraire les relations qui existent entre ces différentes entités. À la fin de ce traitement, le texte sera présenté sous forme de triplets <entité> <relation> <entité>, et sera par la suite exploitable par les technologies du Web sémantique.

Dans notre recherche, nous nous intéressons à la reconnaissance des entités dans le texte aux travers des annotateurs sémantiques, des systèmes qui regroupent deux tâches. Tout d'abord, la détection des entités les plus pertinentes dans le texte et ensuite, leur désambiguïsation en leur affectant des URIs qui permettent ainsi de pointer vers un concept dans une base de connaissances. En effet, une mention textuelle peut être sous forme d'une série de mots comme, par exemple, "Canadian Grand Prix" ou d'un seul mot comme "Montreal". Ces mentions sont la plupart du temps ambiguës. Tout comme le mot "Washington", qui seulement dans Wikipédia peut avoir pas moins de 65 significations différentes (personne, état, ville...). Afin d'identifier la bonne entité, Wikipédia est de loin la source de données la plus utilisée dans ce domaine. Elle est volumineuse, traite d'un grand nombre de sujets (plus de 5 millions

1. https ://www.w3.org/RDF/
2. http ://www.worldwidewebsize.com/

de pages en août 2016) et est gratuite. L'utilisation de Wikipédia a été introduite en 2006, par Bunescu[4], en nommant la tâche "Wikification". Néanmoins, cette méthode reste insuffisante pour atteindre l'objectif du Web sémantique, car Wikipédia est une base de connaissances semi-structurée qui se limite à présenter l'information en HTML, un langage difficilement interprétable par les machines. Toutefois, avec l'apparition du réseau Linked Open Data [3], les annotateurs ont subi une grande évolution, permettant ainsi de pointer leurs annotations vers des bases de connaissances structurées. Elles sont exploitables autant par des humains que par des machines.

Dans ce mémoire, nous nous concentrons, en premier lieu, sur l'amélioration d'un annotateur sémantique, DBpedia Spotlight, vu les avantages qu'il présente et qui seront décrits tout au long de cette analyse. Par la suite, nous exposons l'approche que nous avons mise en place pour améliorer d'autres annotateurs sémantiques soit AIDA, Tagme et Babelfy, et ce, pour notre participation à la compétition Open Knowledge Extraction. Lors de cette compétition, il fallait identifier des types d'entités nommées prédéfinies (organisations, personnes, places et rôles) à partir de petites phrases et créer des annotations qui pointent vers DBpedia, ou générer de nouvelles URI lorsque la mention détectée n'a pas d'entité qui la représente. L'objectif principal, ici, est la population des bases de connaissances, en enrichissant les entités présentes par des informations extraites automatiquement de sources externes. Les questions de recherche auxquelles nous essayons de répondre dans ce contexte sont les suivantes :

— *Quelles sont les limites des annotateurs sémantiques existants ?*
— *Quels sont les bons moyens pour remédier à ces limites ?*

Plan du mémoire

La suite de ce mémoire est organisée de la manière suivante : Le chapitre 2 explore les différentes approches qui ont été développées dans l'état de l'art. Le chapitre 3 contient les méthodes de recherche que nous avons utilisées, ainsi qu'une description plus détaillée de certains concepts abordés dans les chapitres 4 et 5. Les chapitres 4 et 5 représentent respectivement une version intégrale de nos deux articles "FICLONE : improving DBpedia Spotlight using named entity recognition and collective disambiguation" et "Collective disambiguation and Semantic Annotation for entity linking and typing". Enfin, le chapitre 6 contient une discussion générale sur les résultats obtenus dans les deux chapitres précedents. Le chapitre 7, conclut ce mémoire en présentant les différentes améliorations envisageables.

3. http ://linkeddata.org/

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans cette section, nous décrivons ce qu'est un annotateur sémantique ainsi que son architecture globale. Nous présentons par la suite une revue de littérature des systèmes développés dans l'état de l'art en détaillant le fonctionnement interne ainsi que l'utilité de chacune de leurs composantes.

2.1 Définitions générales

Un annotateur sémantique a pour rôle d'extraire les mentions les plus pertinentes dans un texte en entrée et de les lier à des entités dans une base de connaissances existante. Prenons l'exemple suivant :

> *Angelina Jolie, her father Jon, and her partner Brad never played together in the same movie.*

Dans cet exemple, un annotateur sémantique pourrait détecter les six mentions suivantes :

- Angelina Jolie → DBpedia.org/resource/Angelina_Jolie
- father → DBpedia.org/resource/Father
- Jon → DBpedia.org/resource/Jon_Voight
- partner → DBpedia.org/resource/Partnership
- Brad → DBpedia.org/resource/Brad_Pitt
- movie → DBpedia.org/resource/Film

On remarque que chaque mention est liée à un URI qui représente une entité dans la base de connaissances DBpedia. Certains annotateurs sémantiques se contentent d'annoter seulement les entités nommées. Une entité nommée est une mention textuelle qui pourrait être catégorisée. Les catégories qui sont fréquemment utilisées dans le domaine de la recherche d'information sont les personnes, les organisations ou les lieux, mais dans d'autres cas, les dates, les courriels ou même les adjectifs dérivés des noms propres (i.e. American) pourraient aussi être considérés. Dans notre exemple, seules les mentions *Angelina Jolie*, *Jon* et *Brad* sont des entités nommées.

Les annotateurs sémantiques ne doivent pas être confondus avec les systèmes de désambiguïsation lexicale (*word sense disambiguation*). Ces derniers traitent les mots qui ont plusieurs sens. Dans l'exemple précédent, ils s'occupent de trouver le vrai sens du verbe *played* ou $father$. Les mentions traitées sont composées d'un seul mot qui peut être un nom, un adjectif ou un verbe, contrairement aux annotateurs sémantiques qui par exemple extraient

des mentions comme *United State Congress* et où chaque mention doit contenir en moins un nom. Et aussi, comme nous l'avons mentionné, les annotateurs sémantiques utilisent des bases de connaissances dans lesquelles chaque entité est unique (dans DBpedia il y a une seule et unique entité qui décrit l'actrice *Angelina Jolie*). La désambiguïsation lexicale, de son côté, requiert l'utilisation d'un dictionnaire comme *Wordnet* dans lequel un mot peut à son tour avoir plusieurs synonymes.

2.2 Le Web des données

Les annotateurs sémantiques ont subi une grande évolution avec l'émergence du Web des données (Linked Open Data). Le Web des données est un ensemble de bases de connaissances structurées librement accessibles et interreliées par des liens en RDF. Le RDF est un standard de modélisation de données sur le Web, qui permet de décrire des relations entre des entités (qui sont appelées ressources). Une relation en RDF est composée d'un sujet, d'un prédicat et d'un objet. Le sujet et l'objet représentent les ressources, le prédicat est la relation qu'il y a entre elles. DBpedia contient des informations structurées selon les standards du Web de données à partir de Wikipédia. DBpedia est souvent considéré comme la plaque tournante du Web des données, vu le nombre de bases de connaissances qui lui sont liées. DBpedia, YAGO et Freebase sont de plus en plus utilisés dans l'implémentation des annotateurs sémantiques, vue la facilité d'accès à l'information via des requêtes SPARQL. Avec une simple requête SPARQL, nous sommes capables par exemple d'extraire pour l'entité $Brad\_Pitt$, son lieu de naissance, sa femme, le nombre d'enfants qu'il a et la liste des films dans lesquels il a joué.

2.3 Architecture des annotateurs sémantiques

Généralement, les annotateurs sémantiques sont développés selon l'architecture illustrée à la figure 2.1.

La première phase du pipeline consiste à identifier les mentions M les plus pertinentes du texte. Une mention peut être composée d'un ou plusieurs mots. Ensuite, un ensemble de candidats E_m est affecté à chacune des mentions $m \in M$. Un candidat pour une mention donnée m est une entité d'une base de connaissances qui pourrait la représenter, comme le candidat $Washington\_D.C.$ qui fait partie des candidats affectés à la mention $Washington$. Enfin, la phase de désambiguïsation a pour objectif d'identifier le candidat $e \in E_m$ qui est le plus adéquat pour la mention m en tenant compte du contexte dans lequel elle apparaît. Afin de bien décrire chacune des phases que nous venons de citer, nous allons maintenant présenter l'état de l'art pour chacune.

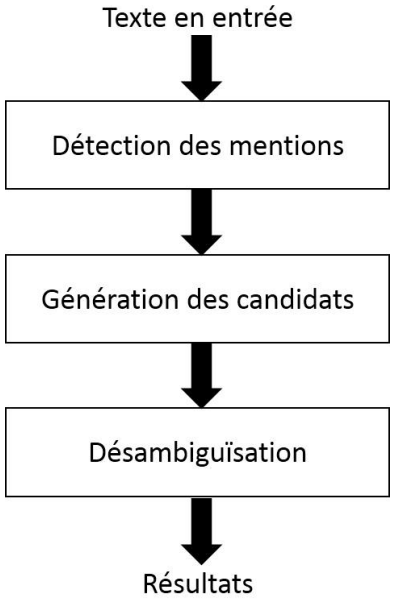


Figure 2.1 Architecture générale des annotateurs sémantiques

2.3.1 La détection des mentions

Cette couche est responsable d'extraire les mentions importantes du texte. Diverses méthodes sont utilisées pour cette tâche. Certains systèmes comme ceux introduits dans [17, 10, 4], optent pour l'utilisation d'un outil de reconnaissance d'entités nommées. Mais comme nous l'avons mentionné, ces systèmes ne traitent pas les concepts ou les mots communs comme *president, father, company*, ce qui limite leur utilisation. La méthode la plus courante dans ce domaine consiste à extraire les expressions ancrées (hyperliens) de Wikipédia. Ces expressions apparaissent dans la description de la plupart des pages Wikipédia pour faire référence à d'autres pages. Elles sont généralement ajoutées manuellement par les internautes, comme dans l'exemple suivant, qui est extrait de la page *wiki/New_York*

> *New York is a* ***state*** *in the* ***Northeastern United States*** *and is the* ***27th-most extensive****...*

Les expressions ancrées dans cet exemple sont **state**, **Northeastern United States** et **27th-most extensive**, qui font référence respectivement aux pages *wiki/U.S._State*, *wiki/Northeastern_United_States* et *wiki/List_of_U.S._states_and_territories_by_area*. Formellement, une expression ancrée est une paire ⟨e,a⟩ , où e est la mention textuelle et a l'hyperlien qui dirige vers l'entité qui décrit cette mention. Ces expressions se voient attribuer un poids, qui correspond à leur fréquence d'occurrence dans Wikipédia, et qui est calculé selon la

formule suivante :

$$fréquence(e) = \frac{N(< e, a >)}{N(e)} \tag{2.1}$$

C'est le nombre de fois où une expression est ancrée, divisé par le nombre de fois que cette expression apparaît dans tout Wikipédia, ancrée ou non. Par exemple, si le mot **state** a été ancré 5000 fois et apparaît 15000 fois dans Wikipédia sans être ancré, dans ce cas fréquence(state)= 5000/(5000+15000)=0,4. Cette fréquence est utilisée lors du traitement d'un nouveau texte, pour ignorer les mots qui ne sont pas importants. Après que cet ensemble de mentions, que nous appelons liste de formes de surface, soit collecté, plusieurs algorithmes sont utilisés pour pouvoir les identifier dans le texte. Dans Babelfy [23], les auteurs utilisent un étiqueteur morphosyntaxique et extraient tous les fragments du texte (i.e les ensembles de mots successifs de longueur maximum 5), et gardent ensuite ceux qui existent dans la liste de formes de surface. D'autres annotateurs utilisent des implémentations de différents algorithmes comme l'algorithme d'Aho Corasik qui est utilisé dans DBpedia Spotlight [21]. Cet algorithme consiste à identifier les chaînes de caractères dans un texte qui contiennent les mots recherchés. Ces mots recherchés sont l'ensemble de formes de surface. Ces systèmes utilisent souvent un étiqueteur morphosyntaxique pour éliminer les mentions qui sont composées seulement de verbes, d'adverbes ou d'adjectifs. Une autre méthode a été utilisée dans NERSO [15], qui consiste à utiliser des requêtes SPARQL sur DBpedia pour identifier les mentions pertinentes. Les auteurs commencent par découper le texte en mots. Ensuite, ils parcourent le texte du début jusqu'à la fin pour identifier les mentions. Tout d'abord, ils prennent les quatre premiers mots, puis ils exécutent différentes requêtes SPARQL. Si ces requêtes retournent un résultat, ces quatre mots seront considérés comme une seule mention puis ils passent aux 4 mots suivants. Si aucun résultat n'est retourné alors ils testent les 3 premiers mots et ainsi de suite.

2.3.2 Génération des candidats

Cette couche est responsable d'identifier, pour chaque mention trouvée par la couche précédente, un ensemble de candidats dans la base de connaissances qui pourraient lui être associés. La méthode la plus utilisée se base entièrement sur Wikipédia en construisant un ensemble de données lexicalisées. C'est un ensemble d'expressions dans lequel chacune est rattachée à un ensemble d'entités qui correspondent à des entrées de Wikipédia. Par exemple, le mot *Washington* est associé aux entrées *Washington,_D.C.*, *George_Washington* et *Washington_(state)*, entre autres. Pour construire un tel ensemble de données, quatre éléments de Wikipédia sont utilisés.

— Les titres des pages Wikipédia : Les titres sont généralement composés d'un ensemble

de mots séparés par des tirets. Par exemple, l'expression *George Washington* aura l'entité George_Washington dans sa liste de candidats.

— Les titres des pages de redirection : par exemple la page de redirection USA qui redirige vers United_States. Dans ce cas, l'expression **USA** est extraite et elle aura comme candidat United_States.

— Les pages de désambiguïsation : comme *wiki/Washington*". Cette page décrit 64 entités qui pourraient bien correspondre à l'expression "Washington". Dans ce cas, l'expression "Washington" est rattachée à ces 64 entités.

— Les expressions ancrées : ce sont les expressions que nous avons définies dans la phase d'identification des mentions. Dans l'étape précédente les expressions sont seulement enregistrées et pondérées par la fréquence d'expressions (formule 2.1) alors que pour la génération des candidats, tous les couples <mot ancré,entité> sont extraits. Chaque entité est par la suite pondérée par la probabilité à priori.

À chaque candidat e trouvé pour une mention m, on peut associer une probabilité à priori. En effet, il s'agit de la probabilité que la mention m soit associée à l'entité e dans Wikipédia. Elle permet d'ordonner les candidats indépendamment du contexte dans lequel elle apparaît. Supposons que nous ayons le mot "Washington" ancré 10 fois dans un corpus donné et qu'il ait été utilisé 5 fois pour faire référence à l'entité **Washington_D.C**, 3 fois pour **George_Washington** et 2 fois pour **Washington_Pennsylvanie**. Alors les probabilités à priori P(Washington_D.C/Washington), P(George_Washington/Washington) et P(Washington_ Pennsylvanie/Washington) seront dans ce cas respectivement 0,5 0,3 et 0,2.

La plupart des systèmes implémentés utilisent cette méthode. Pour chaque mention identifiée, ils utilisent une correspondance exacte pour extraire les candidats. Dans ce cas, les candidats pour l'expression en minuscules **us** ne sont pas forcément les mêmes que pour **US**. YAGO offre la possibilité d'extraire un ensemble de candidats pour une mention donnée, par le biais d'une requête SPARQL sur le prédicat *yago :means*. Par exemple, AIDA [17] utilise ce service offert par YAGO pour générer ses candidats.

Dans Babelfy [23], les auteurs utilisent une approche différente. Pour extraire les candidats, ils utilisent une méthode de *superstring matching*, au lieu de l'*exact matching*, pour pouvoir par exemple extraire l'entité "Mario Gomez" pour la mention "Mario". Donc ils extraient tous les candidats dont le titre contient la mention à traiter.

2.3.3 La désambiguïsation

À la fin de la phase de génération des candidats, chaque mention m qui a été identifiée a maintenant un ensemble de candidats E où chaque $e_i \in E$ correspond à une entité de la

base de connaissances. Ce qui fait vraiment la différence d'un système à un autre, c'est la méthode pour identifier le bon candidat. Afin de bien présenter chaque approche, nous allons les classer en trois catégories. Tout d'abord les approches non supervisées, qui utilisent les méthodes classiques de recherche d'information. Puis les approches supervisées, qui utilisent les méthodes d'apprentissage automatique et qui requièrent l'utilisation d'un ensemble de données d'entraînement pour pouvoir fixer la valeur de chaque paramètre utilisé dans le modèle. Et enfin les méthodes basées sur des graphes, où le problème de désambiguïsation est converti en un problème de résolution de graphe dont les nœuds sont généralement les mentions ainsi que leurs candidats.

Les approches non supervisées

Dans ces approches, les entités Wikipédia sont représentées dans un modèle vectoriel, où chaque entité est un point dans un espace multidimensionnel des mots. Ce qui veut dire que chaque entité e_i est représentée par un vecteur de mots, que certains auteurs appellent vecteur contexte v(e_i), et où chaque mot peut être pondéré, selon une des techniques que nous expliquons plus loin. Les mots w du contexte d'une entité e_i sont généralement extraits de leur page Wikipédia. Ce qui veut dire que l'ensemble V des entités de Wikipédia est représenté sous la forme suivante :

$$V = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{pmatrix} \qquad o\text{ù} \qquad v_i \begin{pmatrix} w_{i,0} \\ w_{i,1} \\ \vdots \\ w_{i,m} \end{pmatrix} \tag{2.2}$$

où $w_{i,j}$, $j \in [0..m]$ est un mot du vecteur contexte $v(e_i)$. Le TF-IDF est une méthode typique de pondération des mots en recherche d'information. Il s'agit d'une combinaison de deux valeurs : TF et IDF. Le TF d'un mot du contexte w d'une entité e_i (i.e $w \in v(e_i)$), que nous notons par TF(w), est la fréquence d'occurrence de ce mot divisée par le nombre total de mots dans le contexte. Alors que TF reflète l'importance d'un mot dans un contexte donné, le IDF quant à lui reflète l'importance de ce mot dans un corpus entier, ce qui dans notre cas comprend tous les mots des contextes de toutes les entités e_i de Wikipédia. Le IDF d'un mot est donné par la formule suivante :

$$IDF(w) = log\frac{N}{|\{v \in V : w \in v\}|} \tag{2.3}$$

où N=|V|=|E| est le nombre total d'entités Wikipédia et $|\{v \in V : w \in v\}|$ le nombre de

vecteurs contextes dans lesquels le mot w apparaît. Par la suite, le TF-IDF d'un mot dans un vecteur contexte v donné est calculé par la formule suivante :

$$TF - IDF(w, v) = TF(w, v) * IDF(w) \tag{2.4}$$

DBpedia Spotlight est parmi les systèmes qui utilisent cette méthode. Pour chaque mention détectée, il représente son vecteur contexte comme pour les entités de Wikipédia, en prenant les mots à droite et à gauche de la mention. Et pour pondérer les mots du contexte de la mention, que nous notons par v_m, et les mots du vecteur $v(e_m)$ où e_m est une entité candidate pour la mention m, il utilise une variante du TF-IDF, nommée TF-ICF dont la valeur ICF est définie par la formule suivante :

$$ICF(w, V_E) = log\frac{|E_m|}{|\{v \in V_E : w \in v\}|} \tag{2.5}$$

où $|E_m|$ représente le nombre de candidats pour la mention m, et V_E l'ensemble des vecteurs contextes de ces candidats. Ensuite, pour calculer la similarité contextuelle entre v_m et chaque contexte $v \in V_E$, (ce qui revient en quelque sorte à attribuer un score contextuel à une entité $e \in E_m$), il utilise la similarité cosinus qui est donnée par la formule suivante :

$$contextualScore(e) = cos(v_m, v) = \frac{v_m.v}{\|v_m\|.\|v\|} \tag{2.6}$$

À la fin, pour chaque mention m, il retient le candidat e qui a le score le plus élevé, où score(e) est une combinaison de contextualScore(e) et de la probabilité à priori P($e|m$). Cette combinaison est donnée par la formule suivante :

$$combinaison(e) = 1234.3989 * P(e|m) + 0.9968 * contextualScore(e) - 0.0275 \tag{2.7}$$

Dans une autre approche très souvent citée [10], une entité e est représentée par deux vecteurs : le premier contient le contexte et le deuxième, les catégories de cette entité (i.e. les catégories qui existent dans sa page Wikipédia). Elle modélise aussi le document en deux vecteurs, le premier introduisant le contexte de tous les candidats identifiés et le deuxième contenant toutes les catégories qui ont été identifiées dans le texte. Pour désambiguïser chaque mention, ils ont formulé un processus qui tend à maximiser la concordance entre le vecteur contexte d'une entité donnée avec celui du document, et de même pour le vecteur de catégories.

Les approches supervisées

Les premiers à avoir utilisé l'apprentissage automatique pour la désambiguïsation étaient aussi les premiers à avoir utilisé Wikipédia comme base de connaissances pour faire référence aux mentions traitées [4]. Ils ont utilisé la méthode de similarité cosinus comme dans DBpedia Spotlight. Mais ils ont constaté que cette méthode n'est pas suffisante pour traiter des textes courts. Pour cela, chaque entité est représentée par deux vecteurs, un vecteur contexte v et un vecteur de catégories c. Leur ajout majeur consiste à entrainer un SVM [19] pour pouvoir calculer la corrélation entre un mot w et une catégorie c_i donnée. Cette méthode leur a servi par exemple pour lier le verbe **sing** avec la catégorie **Musicians**. Ainsi, dans une phrase comme *Robbie Williams sings*, cela leur permet d'identifier le fait qu'il s'agit bien du chanteur américain et non pas du joueur de football anglais.

Ce travail a été ensuite poursuivi par [22], où les auteurs commencent par identifier les mentions non ambiguës (i.e celles qui ont un seul candidat e). Ces mentions sont par la suite utilisées comme contexte pour désambiguïser le reste des mentions. Notons par $a_i \in A$ l'ensemble des candidats qui a été assigné pour les mentions non ambiguës. Pour le reste des mentions, les auteurs assignent deux valeurs à chaque candidat. La première est la fréquence de ce candidat, c'est-à-dire le nombre de fois où ce candidat a été utilisé comme destination dans une expression ancrée. Le deuxième paramètre est la moyenne des cohérences du candidat e avec chacune des entités $a_i \in A$. La formule de cohérence a déjà été introduite dans [38] et elle est une adaptation de la formule de distance normalisée de Google [8] :

$$cohérence(a, b) = \frac{log(max(|A|, |B|)) - log(A \cap B)}{log(N) - log(min(|A|, |B|))} \tag{2.8}$$

où a et b sont deux entités de Wikipédia pour lesquelles on veut calculer la cohérence, A et B représentent respectivement les liens rentrants (incoming links) et N le nombre total d'entités qui sont présentes dans Wikipédia. Leur principale contribution est de choisir, pour chaque candidat e_i, entre sa fréquence ou sa cohérence avec le contexte. Et pour cela, ils ont entraîné différents classifieurs (SVM et le Naive Bayes) en utilisant la fréquence et la cohérence pour détecter quand les entités $a_i \in A$ du contexte sont cohérentes entre elles. Dans ce cas les candidats e_i qui ont les cohérences les plus élevées seront choisis. Sinon, si le texte en entrée ne contient pas de mentions non ambiguës ou s'il est d'ordre général (i.e les a_i ne sont pas cohérentes entre elles), ce sont les fréquences des candidats qui seront utilisées.

Une amélioration à cette idée a été implémentée dans Tagme [13]. Contrairement au système précédent, pour désambiguïser une mention m les auteurs ont utilisé toutes les entités candidates $e_a \in E_a$ des mentions $a \neq m$. Et pour cela, ils ont mis en place un système de vote

défini par la formule suivante :

$$vote_a(e_m) = \frac{\sum_{e_a \in E_a} rel(e_a, e_m).Pr(e_a|a)}{|E_a|} \tag{2.9}$$

où e_m est un candidat pour la mention m, $rel(e_a, e_m)$ est la cohérence entre deux candidats et $Pr(e_a|a)$ est la probabilité à priori. Cette formule représente le vote des candidats d'une seule mention $a \neq m$ pour que e_m soit le bon candidat pour m. Par la suite, ils ont affecté pour chaque candidat e_m le paramètre $rel_m(e_m)$, la somme des votes reçus par les différentes mentions, $rel_m(e_m) = \sum_{a \neq m} vote_a(e_m)$. Pour la désambiguïsation les auteurs ont utilisé un classifieurs à deux paramètres $rel_m(e_m)$ et $Pr(e_m|m)$. Ce classifieur a pour rôle de choisir le candidat qui a le score de classification le plus élevé. Une fois l'étape de désambiguïsation achevée, les auteurs ont ajouté une méthode supplémentaire pour éliminer les mauvaises annotations et améliorer la précision de leur système. Notons A le candidat qui est affecté à la mention a. La fonction qu'ils ont définie combine la moyenne de la cohérence de A avec les autres annotations et la fréquence de l'expression $fréq(a)$ (formule 2.1). Si ce score est inférieur à un seuil ρ_{NA}, l'annotation est éliminée.

Les méthodes basées sur des graphes

L'idée ici consiste à convertir la tâche de désambiguïsation en un problème de résolution de graphe, dans lequel les nœuds représentent généralement les mentions ainsi que leurs candidats.

Dans NERSO [15], les auteurs utilisent un graphe orienté pour représenter les différents candidats identifiés. En se basant sur les liens entrants et les liens sortants, ils ont utilisé la méthode de centralité des graphes [34] pour pondérer chaque entité. Pour chaque mention m, ils retiennent le candidat e qui a obtenu la pondération la plus élevée. Mais la limite de cette approche est qu'elle se base seulement sur les liens entrants et les liens sortants : le contexte dans lequel apparaît chaque mention est ignoré.

Dans AIDA [17], les auteurs ont utilisé un graphe simple dont les nœuds représentent les mentions et les candidats. Deux types d'arcs sont pondérés : les arcs de type mention-entité, qui sont pondérés par une combinaison entre la similarité cosinus des contextes et la probabilité à priori, et les arcs de type entité-entité, dont le poids constitue la cohérence entre ces entités (formule 2.8). Ils implémentent la désambiguïsation comme un problème de sélection du sous-graphe permettant de maximiser une formule qui combine la probabilité à priori, la similarité des contextes et la cohérence entre les entités.

Pour Babelfy [23], les auteurs ont présenté une méthode qui sert à la fois à la désambiguïsation lexicale et l'annotation sémantique. Leur idée consiste à appliquer les méthodes généralement utilisées pour la désambiguïsation lexicale au cas spécifique des entités nommées et vice versa. Ils ont utilisé Babelnet [24] comme base de connaissances, ce qui permet à leur système d'être multilingue. Babelnet est un réseau qui regroupe Wikipédia et Wordnet. Chaque sommet de ce réseau contient soit un concept, soit une entité nommée, qui sont liés entre eux par différentes relations sémantiques. L'équipe de Babelnet a tout d'abord commencé par traiter ce réseau sémantique, en affectant à chaque sommet v ce qu'ils appellent une signature sémantique $semSign_v$. À partir de chaque sommet v, ils ont appliqué le principe de marche aléatoire pour découvrir d'autres sommets $v' \neq v$. Ces sommets forment ainsi la signature sémantique du sommet v. Ensuite, ils modélisent le texte en un graphe orienté, dont les sommets sont sous la forme $< C, M >$, où C est un candidat pour une mention M donnée. Un arc est ajouté de $< C, M >$ vers $< C', M' >$ si et seulement si, $C' \in semSign_C$ et $M \neq M'$. Chaque sommet est par la suite pondéré par un score faisant intervenir le nombre d'arcs rentrants et sortants. Enfin, pour chaque mention M, ils choisissent le candidat qui a le score le plus élevé. L'avantage de Babelfy est qu'il permet de limiter les entrées désambiguïsées à des entités nommées seulement, les sens des mots ou bien les deux.

xLisa [39] est un nouveau système qui, comme [23], permet d'annoter différentes langues. Contrairement à AIDA [17], il utilise un graphe orienté, dans lequel chaque mention est directement liée à son ensemble de candidats. Les arcs mention-entité sont aussi pondérés par la similarité contextuelle avec la probabilité à priori et les arcs mention-entité, par la formule de cohérence (formule 2.8). Ils ont par la suite utilisé l'algorithme de $PageRank$ personnalisé introduit par [1], pour pouvoir attribuer à chaque nœud une probabilité qui résulte des différents paramètres présents dans le graphe. Pour la désambiguïsation, ils gardent le nœud qui a la probabilité maximale.

Il est rare que les chercheurs dans le domaine prennent un système existant pour essayer de l'améliorer. Cependant, il existe quelques améliorations qui ont été effectuées.

WAT [28] représente une amélioration de Tagme, dans lequel les auteurs améliorent la phase de repérage des mentions en ajoutant un autre paramètre pour identifier si la mention a du sens. L'algorithme original a été transformé en un problème de résolution de graphe dans lequel les nœuds sont les mentions et les candidats. Pour la relation mention-entité, la pondération combine la probabilité à priori et la similarité du contexte calculé par la méthode BM25 qui est fournie par Lucene. Et pour la relation entité-entité, ils utilisent le Jaccard [18], plutôt que la formule de cohérence 2.8. L'objectif de leur approche est de trouver le sous-graphe qui interconnecte le plus les mentions du graphe principal.

Dans une autre proposition[31], les auteurs améliorent Wikipédia Miner en changeant la formule de cohérence par celle de Jaccard [18] et ils reportent une amélioration de 8 points en F1-Score. Une amélioration de DBpedia Spotlight a été introduite dans [27], où l'auteur s'est concentré sur des types de mentions prédéfinis (i.e des personnes, place, organisation, produit ou travail créatif). L'auteur entraine un classificateur sur un corpus donné pour identifier les mentions NIL (i.e les mentions qui n'ont pas d'entités qui les représentent dans la base de connaissances). Ensuite, son approche procède au changement des mots du texte en entrée (des majuscules au minuscules), puisque la phase de repérage des mentions est sensible à la casse, pour générer plus de mentions et plus de candidats (par exemple, DBpedia Spotlight génère un candidat différent pour PARIS et Paris). L'auteur a conclu que leur méthode règle certaines limites de Spotlight, comme l'identification des mentions NIL, mais elle introduit aussi du bruit dans leurs résultats.

2.4 Mise en contexte

Tagme, Babelfy, DBpedia Spotlight et AIDA sont parmi les systèmes d'annotation sémantiques les plus cités. Ils sont couramment comparés les uns aux autres. L'annotation sémantique elle-même, que nous venons de décrire, ne se limite pas seulement aux entités nommées, mais aussi aux autres concepts. Pour faire une bonne comparaison entre des annotateurs sémantiques, il faut tout d'abord s'assurer que l'ensemble de données de tests soit compatible avec la tâche de chacun de ces annotateurs. Au cours de nos recherches, nous avons remarqué que ce n'est pas toujours le cas. Dans tous les ensembles de données de tests utilisés pour la comparaison, seulement les entités nommées sont annotées. AIDA et Babelfy se focalisent essentiellement sur l'annotation des entités nommées, ce qui n'est pas le cas pour DBpedia Spotlight et Tagme. Ces derniers sont alors les seuls annotateurs sémantiques accessibles, qui permettent d'annoter plusieurs types de mentions. Nous décrivons dans ce mémoire pourquoi nous avons préféré DBpedia Spotlight à Tagme, en premier lieu. Nous présentons par la suite deux améliorations de DBpedia Spotlight. Dans la première, nous montrons l'importance d'utiliser un outil de reconnaissance d'entités nommées pour que la comparaison avec les systèmes de désambiguïsation d'entités nommées ait plus du sens. Dans la seconde amélioration, nous montrons une approche différente qui combine les sorties de DBpedia Spotlight avec celles de l'outil de reconnaissance d'entités nommées pour pouvoir détecter les concepts qui ne sont pas gérés par la première amélioration. Nous montrons dans cette deuxième amélioration l'intérêt d'ajouter un étiqueteur morphosyntaxique pour augmenter la précision de notre système. Enfin, nous présentons le système WESTLAB avec lequel nous avons remporté la

compétition OKE challenge [1].

1. github.com/anuzzolese/oke-challenge-2016

CHAPITRE 3 MÉTHODOLOGIE

Ce chapitre consiste à présenter les méthodologies que nous avons suivies pour le développement des deux systèmes FICLONE et WESTLAB qui sont décrits respectivement dans les chapitres 4 et 5. Durant nos recherches, nous avons réalisé trois différents types d'expérimentations. Nous nous sommes concentré en premier lieu sur les entités nommées en mettant au point des méthodes adéquates pour améliorer les trois phases de l'annotation sémantique, à savoir la détection des mentions, la génération des candidats et la désambiguïsation. Ensuite, nous avons étendu notre amélioration à la tâche globale d'annotation sémantique pour retourner les concepts ainsi que les entités nommées pertinentes. Enfin, la dernière expérimentation que nous présentons est celle de la compétition OKE, à laquelle nous avons participé et que nous avons remportée, dans laquelle les types d'entités nommées étaient prédéfinis.

3.1 Amélioration de DBpedia Spotlight avec les services FICLONE

Dans cette section, nous commençons tout d'abord par justifier pourquoi nous avons choisi d'utiliser DBpedia Spotlight [1]. Nous présentons par la suite ses erreurs les plus fréquentes, pour ensuite décrire les différentes approches que nous avons implémentées pour les régler, que ce soit pour le cas spécifique des entités nommées ou pour la tâche d'annotation sémantique d'une manière générale.

3.1.1 Le choix de DBpedia Spotlight

Nous avons préféré utiliser DBpedia Spotlight plutôt que Tagme pour plusieurs raisons. Premièrement, le paramètre de confidence qu'offre DBpedia Spotlight nous a aidé à détecter les mentions que DBpedia Spotlight a du mal à traiter. Deuxièmement, le service "Candidates", que nous décrivons en détail plus loin, nous permet de savoir à quel niveau DBpedia Spotlight s'est trompé pour annoter une mention, bien que cette dernière soit bien identifiée. Tagme n'offre pas cette possibilité. Il se contente de renvoyer l'ensemble d'annotations qu'il est capable de traiter. Finalement, le chemin que nous voulions suivre dès le début dans nos recherches était d'étudier l'impact que pourrait avoir une méthode de désambiguïsation collective sur un annotateur sémantique. Comme nous l'avons décrit dans la section précédente, Tagme implémente déjà la désambiguïsation collective par le biais d'un système de vote. DBpedia Spotlight, quant à lui, prend une décision pour une mention donnée indépendamment

1. http ://spotlight.sztaki.hu :2222/rest/annotate

des décisions qu'il va prendre pour les autres mentions. Pour bien mener nos recherches, nous avons commencé par identifier les limites de DBpedia Spotlight, qui sont présentées dans la section suivante.

3.1.2 Les limites de DBpedia Spotlight

La détection des erreurs de DBpedia Spotlight était une étape cruciale dans notre projet, pour savoir quelle partie on devait améliorer en premier. Afin de mieux les analyser, nous commençons par décrire son mécanisme.

DBpedia Spotlight offre un paramètre appelé $Confidence$, qui permet aux utilisateurs de choisir entre la précision et le rappel selon le besoin. Ce paramètre varie entre 0.0 et 1.0. Une valeur très basse permet d'augmenter le rappel, tandis qu'une valeur élevée favorise plutôt la précision. Pour une valeur de 1, si les deux meilleurs candidats ont des scores très proches, ce qui signifie que la mention est très ambiguë, la mention est enlevée du résultat final. Contrairement à 0.0, qui retourne pour chaque mention détectée le candidat qui a le score le plus élevé sans tenir compte du fait que le deuxième candidat a un score proche ou pas.

Vu que la valeur de confidence recommandée est 0.5, notre première idée était de découvrir quelles sont les raisons qui ont poussé DBpedia Spotlight à éliminer des mentions qu'il a identifiées à une valeur de confidence de 0.0. Pour cela, nous avons commencé par détecter les erreurs générées à 0.5 puis celles à 0.0.

Les erreurs de reconnaissance des mentions

Les exemples d'erreurs à une valeur de confidence de 0.5 sont montrés dans la figure 3.1.

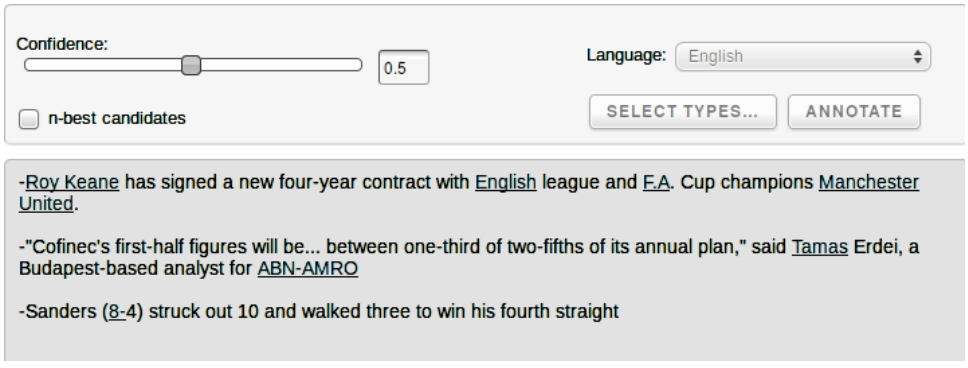


Figure 3.1 Exemples d'erreurs de reconnaissance des mentions de DBpedia Spotlight 0.5

Dans la figure 3.1, nous remarquons qu'il y a trois types d'erreurs lors de la phase de détection des mentions.

— La première phrase montre que DBpedia Spotlight pose un problème avec les points, la bonne mention étant *F.A. Cup champions*, mais la présence du point au milieu pousse Spotlight à retourner seulement la première partie.

— Pour la deuxième phrase, c'est un autre type d'erreur. DBpedia Spotlight n'est pas capable de gérer les mentions pertinentes qui n'ont pas d'entités qui les représentent dans DBpedia (i.e *Tamas Erdei*). Dans ce cas, Spotlight retourne seulement la première moitié *Tamas*.

— Quant à la troisième phrase, c'est le problème des chiffres. Dans cet exemple, DBpedia Spotlight retourne "8-" et l'annote par *DBpedia.org/resource/8-bit*, ce qui n'a aucun rapport avec le contexte de la phrase.

Des exemples d'erreurs supplémentaires sont identifiés à la confidence 0.0. La figure 3.2 en donne quelques exemples.

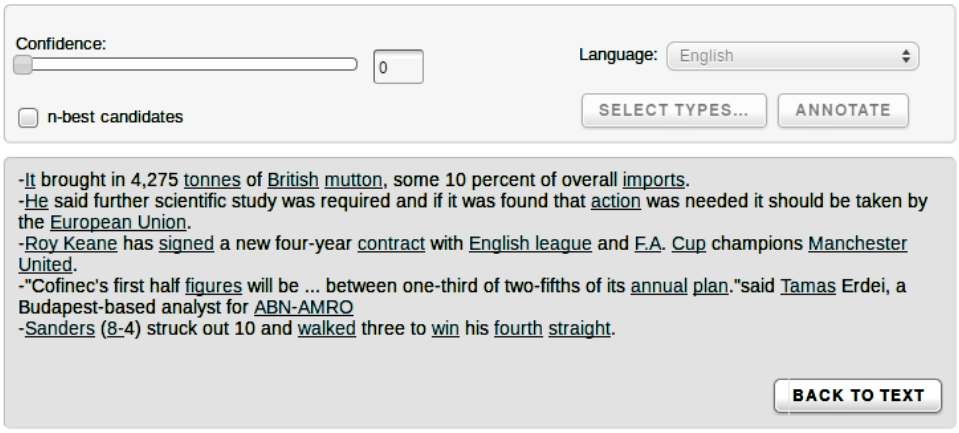


Figure 3.2 Exemples d'erreurs de reconnaissance des mentions de DBpedia Spotlight 0.0

Dans l'article qui présente DBpedia Spotlight [21], les auteurs indiquent qu'ils utilisent un étiqueteur morphosyntaxique pour ignorer les mots composés seulement de verbes, adverbes ou adjectifs. Toutefois, dans les différentes phrases présentées dans la figure 3.2, nous remarquons que les mots "signed, walked" ont été annotés, tout comme les pronoms personnels.

Les erreurs de la phase de génération des candidats

Le module de génération de candidats sert à affecter à chaque mention une liste d'entités de DBpedia qui pourraient bien la décrire. DBpedia Spotlight offre une fonction "Candidates" qui retourne les N meilleurs candidats qu'il utilise pour la phase de désambiguïsation. C'est

une fonction que nous avons jugée importante pour l'implémentation de notre algorithme, vu que les candidats sont ordonnés selon les scores que Spotlight calcule, à savoir le score contextuel et la probabilité à priori. Nous avons effectué une série de tests pour savoir si cette fonctionnalité est suffisante et si elle a des limites. Pour que DBpedia Spotlight soit performant et rapide, la liste des candidats qu'il gère pour chaque mention m a été limitée aux N candidats qui ont les probabilités à priori les plus élevées. Le problème est que cette méthode génère en contrepartie plusieurs erreurs. Nous décrivons l'une des erreurs dans la figure 3.3

Figure 3.3 Exemple d'erreur de génération de candidats DBpedia Spotlight

Bien qu'à la confidence 0.5 la mention PARIS soit ignorée, à 0.0 nous comprenons que c'est dû au fait que le bon candidat n'existe pas dans la liste. Cette erreur est générée parce que la bonne ressource "dbpedia.org/resource/Paris" a une probabilité à priori P(e/PARIS) plus faible que celle de Morse_code et Paper_Aircraft_Released_Into_Space. C'est dû aussi au fait que DBpedia Spotlight utilise une recherche qui est sensible à la casse pour générer les candidats. Nous avons essayé de changer la mention en "Paris", et dans ce cas DBpedia Spotlight arrive à bien identifier le bon candidat.

Nos objectifs de recherches étaient de rendre DBpedia Spotlight plus compétitif par rapport aux annotateurs existants, sans perdre son meilleur atout, qui est l'annotation sémantique qui va au-delà de la reconnaissance des entités nommées. Pour cela, nous avons développé deux variantes de DBpedia Spotlight sous forme de services Web. La première est FICLONE NED [2] dans laquelle nous nous sommes focalisé uniquement sur les entités nommées, alors que la deuxième, FICLONE SA [3], est une version différente qui retourne chaque mention pertinente dans le texte, incluant celles qui ne sont pas des entités nommées.

2. http ://westlab.polymtl.ca/service1/FICLONE/annotate/NED
3. http ://westlab.polymtl.ca/service1/FICLONE/annotate/SA

3.1.3 Amélioration de Spotlight pour la désambiguïsation d'entités nommées (FICLONE NED)

Pour faire de DBpedia Spotlight un meilleur NED, nous avons remédié aux problèmes décrits ci-dessus, étape par étape. Nous avons commencé par l'identification des mentions, puis la génération des candidats pour enfin implémenter un algorithme de désambiguïsation collective.

L'amélioration de la reconnaissance des mentions

Nous avons conduit une série de tests pour améliorer ce module. Nous avons détecté qu'introduire un outil de reconnaissance d'entités nommées était primordial pour deux raisons :

— Un outil de reconnaissance d'entités nommées est indépendant des bases de connaissances. Il est généralement basé sur des règles morphosyntaxiques et des méthodes d'apprentissage automatique. Cet outil nous offre la possibilité de reconnaître les entités qui ne sont pas présentes dans une base de connaissances donnée.
— Il permet aussi d'éliminer les mentions retournées par l'annotateur sémantique qui ne sont pas des entités nommées, à savoir les verbes, les adjectifs ainsi que les numéros.

C'est pour ces raisons que nous avons opté pour l'utilisation d'un outil de reconnaissance d'entités nommées. Nous nous sommes fiés aux diverses études comparatives effectuées dans ce domaine [2, 11] pour choisir Stanford NER [14].

L'amélioration du générateur de candidats

À cette étape, l'ensemble des mentions que nous avons traitées peuvent être groupées selon deux catégories. La première catégorie M_D, représente les mentions qui sont retournées par Stanford NER ainsi que DBpedia Spotlight avec le paramètre de confidence à 0.5, tandis que la deuxième catégorie M_A, contient les mentions qui sont retournées seulement par Stanford NER, mais qui pourraient contenir des mentions qui sont identifiées par DBpedia Spotlight 0.0. Notre approche, qui est décrite dans le chapitre 4, repose entièrement sur les mentions désambiguisées de M_D pour traiter l'ensemble des mentions de M_A. Pour ces mentions, nous avons directement pris les annotations retournées par DBpedia Spotlight. Puisque la liste de candidats retournés par DBpedia Spotlight 0.0 ne contient pas toujours le bon candidat, et que les mentions qui sont retournées seulement par Stanford NER n'ont pas encore de candidats, nous étions dans l'obligation d'identifier une nouvelle source de candidats. Le premier générateur de candidats que nous avons développé repose entièrement sur des informations de DBpedia accessibles via des requêtes SPARQL en trois étapes (figure 3.4). Étant donné

m la mention pour laquelle nous voulons générer un ensemble de candidats :

```
SELECT ?entite WHERE
{
?entite rdfs:label "m"@en
}

SELECT ?original WHERE
{
?original dbo:wikiPageRedirects ?entite
}

SELECT ?entiteFinal WHERE
{
?original dbo:wikiPageDisambiguates ?entiteFinal
}
```

Figure 3.4 Requetes SPARQL

1. La première requête sert à retourner toutes les entités qui ont comme `rdfs:label` la mention m voulue
2. La deuxième vérifie chacune des entités retournées par la première requête pour savoir si elles correspondent à une page de redirection. Si oui, alors on retourne la page originale sinon, on garde l'entité initiale.
3. La troisième vérifie si une entité est une page de désambiguïsation. Si c'est le cas, alors elle retourne toutes les entités qu'elle contient à la deuxième requête. Les requêtes deux et trois sont répétées jusqu'à ce qu'il n'existe plus de pages de désambiguïsation dans les résultats.

Cette première méthode présentait beaucoup de limites. Pour la mention *Canada* par exemple, cette méthode permet de retourner seulement le pays, mais nous avions aussi besoin des équipes sportives, des sociétés qui contiennent le mot Canada, etc. En plus, les requêtes SPARQL que nous venons de décrire sont sensibles à la casse. Si au lieu de chercher la mention *Canada*, nous cherchons la mention *canada*, cela ne retournera aucun résultat. Cependant, le temps d'exécution d'une requête sur DBpedia qui est insensible à la casse est très grand, ce qui est inacceptable vu que la phase de génération des candidats n'est qu'une phase intermédiaire pour passer ensuite à la phase de désambiguïsation. Pour remédier à ce problème nous avons trouvé un ensemble de données, décrit dans [3], dans lequel chaque ligne contient une mention et une entité qui la représente. Les sources de ces mentions sont les titres des pages DBpedia, les redirections, les pages de désambiguïsation ou les expressions ancrées de Wikipédia. Les auteurs présentent différents ensembles de données. Nous avons

choisi d'utiliser celui qui est le plus favorable à la précision vu qu'aucun score n'est attribué à la mention (i.e la probabilité à priori n'est pas retournée). Nous avons modifié toutes les mentions pour les rendre en minuscules pour que la recherche soit insensible à la casse. Nous avons obtenu à la fin de ce traitement des lignes sous la forme m_i : e_{i0} ; ;e_{i1} ; ;e_{i2}...e_{in}, où m_i est une mention donnée et e_i l'ensemble de candidats qui lui est relié. Ces données ont ensuite été indexées par Lucene pour garantir un accès rapide. Nous avons aussi remarqué que dans cet ensemble de données, daté de 2014, les URI qu'il contient peuvent avoir changé en redirections ou même en des pages de désambiguïsation. Pour ces raisons, nous avons conservé les requêtes SPARQL 2 et 3 à la suite de la génération de candidats telle que décrite ci-dessus.

La désambiguïsation

Ce module consiste à identifier le meilleur candidat pour chaque mention, en se basant sur les annotations qui sont retournées par DBpedia Spotlight 0.5. Nous avons mis en place un algorithme qui se base sur les entités rentrantes et sortantes de chaque candidat e_i pour une mention donnée m_i avec les entités e_j des mentions m_j qui ont déjà été annotées par DBpedia Spotlight. Les détails de notre algorithme ainsi que les expérimentations que nous avons conduites seront décrits dans le chapitre 4. La figure 3.5 présente un exemple d'une entité rentrante (Saputo_Stadium), l'entité sortante (Montreal) et de l'entité à traiter (Montreal_Impact).

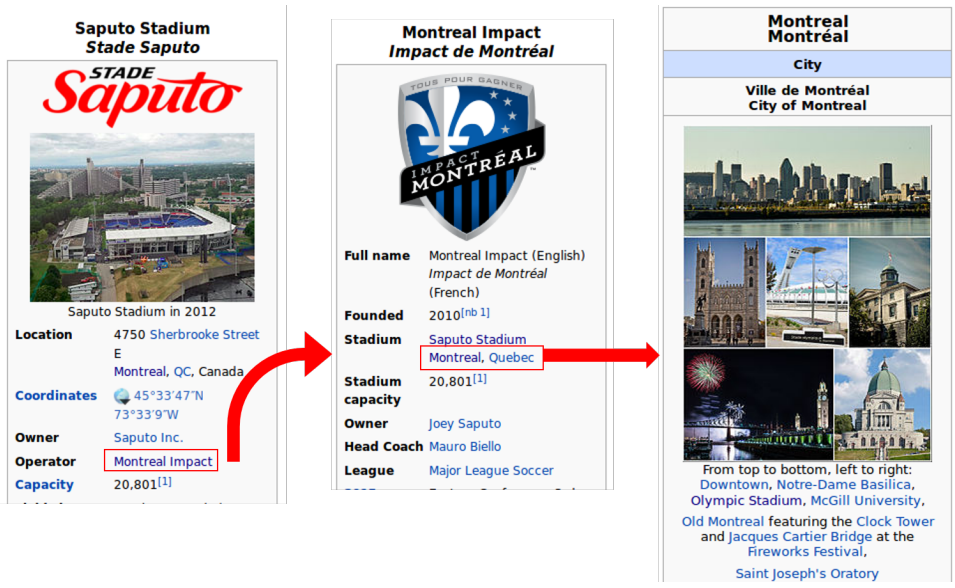


Figure 3.5 Description des entités rentrantes et sortantes

Présentation des données de test NED

Pour évaluer notre système et le comparer aux meilleurs NED, nous avons utilisé 5 différents ensembles de tests de l'état de l'art, dans lesquels seules les entités nommées ont été annotées :

- AIDA/ConLL Complet [4] [17] : ces données ont été produites par les développeurs de AIDA. Ce corpus contient 3393 nouvelles provenant de "Reuters News Stories".
- KORE50 [5] [16] : ce corpus contient 50 phrases, provenant de AIDA/ConLL, offert également par l'équipe de AIDA. C'est un ensemble de petites phrases qui contient des entités nommées très ambiguës.
- MSNBC [6] est une collection de 20 textes qui sont des nouvelles extraites de MSNBC, ce corpus a été introduit dans [10], où seulement les mentions les plus pertinentes sont annotées.
- N3 RSS [7] : ce corpus est introduit dans [32]. Les auteurs ont construit un système pour extraire des flux RSS à partir des sites des journaux les plus connus du monde. Ensuite, ils ont choisi aléatoirement 500 d'entre eux et ils les ont annotés.
- N3 reuters 128 [8] [32] : contient 128 phrases extraites de "Reuters News Stories". Les entités nommées annotées sont des organisations, des personnes ou des endroits.

3.1.4 Amélioration de Spotlight comme annotateur sémantique (FICLONE SA)

La différence majeure avec la version NED est principalement au niveau de l'identification des mentions. L'idée que nous avons implémentée fusionne les mentions retournées par Stanford NER avec celle de DBpedia Spotlight 0.0. Nous avons rencontré certains cas où deux mentions de sources différentes ont un mot ou plus en commun. Par exemple "England" qui est retourné par Stanford NER et la mention "England football team", retourné par DBpedia Spotlight. Dans ce cas, nous gardons toujours la mention la plus longue.

Puisque nous avons gardé toutes les mentions retournées par DBpedia Spotlight dans cette version, nous avons aussi implémenté une approche supplémentaire pour régler les erreurs décrites dans la section 3.1.2. Nous avons utilisé un étiqueteur morphosyntaxique (Stanford POS Tagger) pour garder les mentions qui contiennent au moins un nom. Une telle approche nous a permis d'augmenter remarquablement la précision du système au niveau de l'identifi-

4. mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/aida/downloads/
5. mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/aida/downloads/
6. cogcomp.cs.illinois.edu/page/resource_view/4
7. github.com/AKSW/n3-collection
8. github.com/AKSW/n3-collection

cation des mentions. En ce qui concerne la désambiguïsation, les mentions qui sont retournées par Stanford NER sont traitées de la même manière que la version NED. Nous n'avons apporté aucune modification pour l'annotation des mentions qui sont seulement retournées par DBpedia Spotlight.

Présentation des données de test SA

Pour l'annotation sémantique, nous avons utilisé deux corpus différents qui comprennent des annotations qui ne sont pas des entités nommées uniquement.

Le premier est celui qui est fourni par l'équipe de DBpedia Spotlight [9]. Ce corpus est formé de 58 phrases qui ont été extraites à partir du "New York Times" et qui ont été annotés manuellement. Ce corpus comporte 330 annotations.

Le deuxième corpus est fourni par l'équipe de Tagme [10]. Il se compose de 180000 textes, qui sont à l'origine des fragments de Wikipédia. Prenons l'exemple suivant :

> Phrase : the United States Census Bureau, the town has a total area of 0.6 km^2 (0.2 mi^2), all land. Demographics. As of the census of 2000, there were
> Annotations : km^2 72760 | demographics 3434750 | the united states 3434750 | mi^2 88945 | united states census bureau 57070

Comme nous pouvons le remarquer, la première ligne contient la phrase, et la deuxième ligne les annotations. Les mentions ont été annotées par Wikipédia Page ID et non pas par des URI. Puisque ce corpus date de 2009, nous avons d'abord identifié les textes annotés par des Page ID qui existent encore dans la version de 2016, ce qui a réduit le nombre de textes de 180000 à 172473 textes. Nous avons par la suite choisi les 10000 premiers textes, ce qui a conduit à avoir un corpus qui contient 47554 annotations.

3.2 Expérimentation dans le cadre du OKE challenge 2016 (WESTLAB)

Présentation de la tâche

La tâche consiste à identifier des entités dans une phrase, à les lier à un URI DBpedia (s'il existe) avec des méthodes de désambiguïsation, puis à affecter à chaque mention retournée un type à partir d'un ensemble de types donnés à l'avance. Les types sont : organisation, personne, place ou rôle.

9. yovisto.com/labs/ner-benchmarks/
10. acube.di.unipi.it/tagme-dataset/

Ensemble de données

L'ensemble de données qui a été utilisé lors de la compétition se compose de 55 phrases qui contiennent 86 rôles, 105 organisations, 105 personnes et 44 places. De ces entités, 288 sont représentées par une entité de DBpedia, et 52 ne le sont pas (i.e. elles sont reliées à une mention NIL).

Exemple : *Nicolaas Bloembergen received his Ph.D. degree from University of Leiden under Cor Gorter in 1948 ; while pursuing his PhD at Harvard, Nicolaas Bloembergen also worked part-time as a graduate research assistant for Edward Mills Purcell at the MIT Radiation Laboratory.*

L'annotation est présentée dans le tableau 3.1.

Comme le montre cet exemple, il y a quelques différences avec la tâche qu'effectue FICLONE NED. La première est l'ajout du typage et la restriction des entités nommées à ces types particuliers. La deuxième différence est l'ajout de la résolution de coréférence qui était nécessaire pour cette tâche (i.e *he, she, his, her ...*).

Présentation de l'approche

Notre approche pour OKE (le système WESTLAB) est un peu différente de celle que nous avons établie pour améliorer les performances de DBpedia Spotlight (le système FICLONE NED et SA). Dans WESTLAB, nous avons amélioré quatre annotateurs sémantiques existants, soit DBpedia Spotlight, Tagme, Babelfy et AIDA, pour ensuite utiliser celui qui obtenait les meilleurs scores, à savoir Babelfy. L'utilisation de Stanford NER n'était pas suffisante, vu qu'il n'est pas capable de retourner des mentions de type "rôle" (ce n'est pas considéré comme une entité nommée). Nous avons aussi implémenté une méthode qui fusionne les men-

Tableau 3.1 Annotation pour le OKE challenge

mention	URI	Type
Nicolaas Bloembergen	Nicolaas_Bloembergen	Person
University of Leiden	Leiden_University	Organization
Cor Gorter	Cornelis_Jacobus_Gorter	Person
his	Nicolaas_Bloembergen	Person
Harvard	Harvard_University	Organization
Nicolaas Bloembergen	Nicolaas_Bloembergen	Person
research assistant	Research_Assistant	Role
Edward Mills Purcell	Edward_Mills_Purcell	Person
MIT Radiation Laboratory	sentence-Radiation_Laboratory_(MIT)	Organization

tions retournées par Babelfy avec celle de Stanford NER. Cette méthode nous a permis, par exemple, de fusionner les deux mentions *State University of New York* et *Cortland* et en faire une seule mention qui est *State University of New York at Cortland*. Nous avons aussi intégré un analyseur morphosyntaxique [36] pour filtrer les verbes ainsi qu'un système de résolution de coréférence [20] pour ajouter les pronoms personnels.

Pour la désambiguïsation, l'algorithme implémenté est semblable à celui implémenté pour FICLONE NED. Il se base essentiellement sur les annotations retournées par l'annotateur sémantique et les entités rentrantes et sortantes pour chaque entité à traiter.

Après la désambiguïsation, nous avons défini manuellement un ensemble de règles pour affecter à chaque mention un type en utilisant aussi les types retournés Stanford NER. Vu que l'ensemble de types est prédéfini, nous avons filtré les mentions pour garder seulement celles qui nous intéressent. C'est ce module de filtrage par type qui nous a permis d'augmenter la précision de notre système. Le détail du système WESTLAB est présenté au chapitre 5.

3.3 Protocole d'évaluation

Pour l'évaluation de nos systèmes FICLONE et WESTLAB, nous avons utilisé le système GERBIL [11] [37]. Ce dernier a aussi été utilisé pour la compétition. Il contient un grand nombre d'ensembles de données de test. Les métriques utilisées par GERBIL pour évaluer un système sont le rappel, la précision et le F1-Score. Le rappel est le nombre des sorties correctes retournées par le système divisé par le nombre d'annotations existantes dans la référence (Gold Standard) alors que la précision est le nombre des sorties correctes retournées par le système divisé par le nombre total d'annotations retournées par le système. Le F1-score est calculé de la manière suivante :

$$F1 - Score = \frac{2 * Rappel * Précision}{Rappel + Précision} \tag{3.1}$$

L'évaluation des systèmes n'est pas la même pour WESTLAB et FICLONE

L'évaluation de FICLONE

L'évaluation de FICLONE est identifiée par la tâche de A2KB dans GERBIL ; Pour cette évaluation, GERBIL retourne trois scores différents

— le A2KB (le score final) : une sortie est correcte, si et seulement si la mention a bien été extraite et bien désambiguïsée en même temps.

11. gerbil.aksw.org/gerbil/

— La reconnaissance des mentions : une sortie est correcte lorsque la mention a bien été identifiée.
— La désambiguïsation : ici, une sortie est correcte, si et seulement si la mention a bien été extraite et bien désambiguïsée. Ce qui est différent du A2KB est que pour le calcul de la précision, le nombre des sorties correctes retournées par le système est divisé par le nombre de mentions qui ont été bien extraites, ce qui fait que les mentions qui ont été mal extraites n'ont pas d'effet dans cette évaluation.

L'évaluation de WESTLAB

Pour la reconnaissance des mentions, c'est la même que pour la tâche A2KB, une sortie est correcte lorsque la mention a bien été identifiée. Pour les sous-tâches de désambiguïsation et de typage, une sortie est correcte si et seulement si la mention a bien été extraite et bien désambiguïsée, respectivement typée. Le score final est la moyenne des trois F1-scores obtenue pour chaque sous-tâche.

Nous avons présenté dans ce chapitre les méthodologies que nous avons suivies pour implémenter nos deux systèmes FICLONE et WESTLAB, ainsi que les ensembles de tests que nous avons utilisés pour identifier la meilleure configuration de chacune de leurs composantes. Le chapitre 4 présente l'article qui décrit plus en détail FICLONE [5]. L'article présenté dans le chapitre 5 est l'article que nous avons soumis pour participer à la compétition OKE. WESTLAB est le système qui a remporté la compétition [6].

CHAPITRE 4 ARTICLE 1 : FICLONE : IMPROVING DBPEDIA SPOTLIGHT USING NAMED ENTITY RECOGNITION AND COLLECTIVE DISAMBIGUATION

M. Chabchoub, M. Gagnon, and A. Zouaq. FICLONE : improving DBpedia Spotlight using named entity recognition and collective disambiguation, Natural Language Engineering. (soumis).

Les coauteurs de cet article [5] sont mon directeur de recherche Mr. Michel Gagnon et ma codirectrice de recherche Mme. Amal Zouaq. Mes contributions dans cet article sont l'implémentation du système, les différentes expérimentations et la rédaction. Lors du dépôt de ce mémoire, l'article a été soumis au journal *Natural Language Processing*.

Abstract

In this paper we present FICLONE, a system that improves the performance of DBpedia Spotlight, not only for the task of semantic annotation (SA), but also for the substask of named entity disambiguation (NED). To achieve this, we use Stanford NER as a replacement for the spotting phase, in the NED task, and as an additional spotting resource in the SA task. To generate candidates for each mention returned by the NER that is not recognized by DBpedia Spotlight, we use the LRD & WAT dataset proposed in [3], which associates a list of potential entities of Wikipedia to surface forms. We also use a coreference resolution step to filter the candidates for the mentions that refer to a person. Finally, to select the correct entity among the candidates found for one mention, FICLONE relies on collective disambiguation, an approach that has been proved successful in many other annotators, and which takes into consideration the other mentions in the text. As a result, FICLONE is able to annotate mentions that do not exist in DBpedia. Our experiments show that FICLONE not only substantially improves the performances of DBpedia Spotlight for NED subtask, principally against AIDA-ConLL and MSNBC datasets, but also generally outperfoms other state-of-the-art systems. For the SA subtask, FICLONE also outperforms DBpedia Spotlight against the dataset provided by the DBpedia Spotlight team.

4.1 Introduction

The Semantic Web is a web where all the information can be exploited by humans as well as computers. To achieve the transition to the Semantic Web, documents currently on the

Web must be processed to identify the important entities that are mentionned in their text. Semantic annotators are tools widely used to tackle this issue. Their role consists in detecting relevant mentions in input text and linking them to a target knowledge base. Many systems use Wikipedia in their implementation, since it contains a very large volume of information in various subjects. More recently, with the appearance of the Web of Data, and especially the DBpedia knowledge base, most of the annotation services use this highly structured information. One of these services is DBpedia Spotlight, an open-source semantic annotator that is able to detect not only named entities, like *Montreal* and *Winston Churchill*, but also more abstract concepts, like *architecture* and *mayor*. In this paper, this is what we designate by *full semantic annotation*. DBpedia Spotlight has frequently been compared with other annotators that are limited to the disambiguation of named entities (see [9] for example). This task (we will use NED as abbreviation) must not be confused with the full semantic annotation performed by DBpedia Spotlight.

DBpedia has some interesting characteristics that enable the improvement of its performance. In this paper, we introduce FICLONE, an approach that exploits DBpedia Spotlight and obtains better results by using two main features :

— The combination of DBpedia Spotlight with a named entity recognizer.
— The use of a collective disambiguation process, where the decision made for one mention takes into account the decisions made for all the other mentions in the text.

FICLONE is composed of two services, both of them based on DBpedia Spotlight : FICLONE NED [1], which is limited to named entities, and is thus comparable with other NED systems, and FICLONE SA [2], wich performs full annotation.

The rest of this paper is organized as follows. In the next section, we briefly describe the main state-of-the-art systems for SA and NED tasks. In Section 4.3, we describe FICLONE NED and FICLONE SA. In Section 4.4 we provide the results of our evaluation of these two services and show how they compare favorably to the best annotators that are publicly available. In section 4.5, we analyze the outputs of each FICLONE's component. Finally, we conclude in Section 5.5 with an overview of our approach and propose some future works to improve the performance of FICLONE.

1. http ://westlab.polymtl.ca/service1/FICLONE/annotate/NED
2. http ://westlab.polymtl.ca/service1/FICLONE/annotate/SA

4.2 Related work

Several works have been made to tackle the semantic annotation task. In this section we present the most well-known systems in this area. In general, the task can be divided into three main steps : (i) entity spotting, to recognize the most relevant mentions in input text ; (ii) candidate generation, to assign a list of candidates from a knowledge base to every spotted mention ; (iii) entity disambiguation, to find the best candidate for each mention.

DBpedia Spotlight [21] relies on a lexical resource that contains a set of surface forms extracted from DBpedia (titles, redirects) and Wikipedia wikilinks. It uses an implementation of the Aho-Corasick string matching algorithm for spotting and an implementation of Hidden Markov Models to ignore spots that are composed of only verbs, adverbs, adjectives or prepositions. For each surface form in the lexical dataset, there are many associated entities. These entities constitute the candidate list for the mention that corresponds to this surface form. Each of these entities is weighted by a prior score, that represents its probability in Wikipedia (basically, it is the proportion of occurrences of this entity among the total occurrences of all entities associated to the surface form). Due to performance issues, DBpedia Spotlight only considers the best candidates associated to each surface form. DBpedia Spotlight also defines a variant of TF*IDF that is used to calculate the contextual similarity between a mention m and an entity e from the candidate list. For each mention, it keeps the candidate that has the best combination of prior and contextual scores. One limitation of DBpedia Spotlight is that it performs what we call *individual disambiguation*, i.e., every mention is annotated without considering the decision taken for the other mentions.

Recently, *collective disambiguation* approaches have shown much promise to improve the performance of the task. In the following, we describe a set of semantic annotators that rely on collective disambiguation : Wikipedia Miner [22], Tagme [13], AIDA [17], WAT [28] and Babelfy [23]. In collective disambiguation, the disambiguation of a mention has an influence on the disambiguation of the other mentions in the same text. This method was first introduced in Wikipedia Miner [22]. This annotator detects the unambiguous mentions (i.e mentions that have only one candidate) and then uses these annotations to disambiguate the other mentions. It also uses a relatedness formula, which expresses how much two entities are semantically related. Various classifiers (SVM, naive Bayes classifiers and decision trees) are trained to balance between the prior score and the relatedness with other unambiguous mentions.

Another recent semantic annotator based on collective disambiguation, Tagme [13] uses an anchor dictionary that is built and indexed on top of Wikipedia. A mention can be annotated several times. As an example, for *Southern California*, Tagme extracts two different

surface forms : *Southern California* and *California*. Such a method promotes the recall at the expense of precision, since usually there are no overlapping annotations in evaluation datasets. Disambiguation is achieved in a manner similar to Wikipedia Miner, but instead of considering only the unambiguous mentions, all other mentions are used. Let e be a candidate for mention m. For each of the other mentions in the text, a score is determined in relation with entity e. The entity with the highest vote score is then selected. Tagme also implements a pruning phase to detect and erase insignificant spots. WAT [28] is an enhanced version of Tagme, in which two main modifications are made. For the spotting phase, WAT uses the prior score to eliminate non-significant mentions. For the disambiguation step, it represents the disambiguation task as a graph, where mentions and their candidates are represented as nodes. The aim of the approach is to find the sub-graph that interconnects a maximum of mentions from the main graph.

A different approach was developed for AIDA [17], where Stanford NER, which is based on Conditional Random Fields (CRF) models, was used as a named entity recognizer. A list of candidates is produced by searching in YAGO [35] the entities whose label matches the mention's surface form. To disambiguate the spotted mentions, AIDA uses a graph-based algorithm, where both textual mentions and NE candidates are nodes. The mention-candidate edges are weighted with contextual similarity combined with the prior score, while the relatedness defined in Wikipedia Miner is used to weight the candidate-candidate edges. AIDA extracts the sub-graph with the best density, using a combination of the three computed scores (relatedness, prior score and contextual score).

For Babelfy [23], a part-of-speech tagger is used to identify relevant mentions that contain at least one entry in Babelnet [24]. A random walk algorithm is applied to discover the set of entities that are reachable from an entity e. This set of entities is called the "semantic signature" of the entity e. Similarly to AIDA, Babelfy uses a graph-based approach. Every node in the graph is represented by a pair $<m,c>$, where m is a spotted mention and c is one of its candidates. An edge is added from $<m,c>$ to $<m',c'>$ only when $c' \in semanticSignature(c)$ and $m \neq m'$. Each node is weighted with a score that computes the number of its incoming and outgoing edges. At the end, Babelfy keeps, for each mention, the candidate that has the highest score.

4.3 FICLONE

In this section, we describe two new Web services : FICLONE NED, which improves DBpedia Spotlight using collective disambiguation for the named entity disambiguation task, and FICLONE SA, which also exploits DBpedia Spotlight and improves its performance as a

semantic annotator.

4.3.1 Named entity disambiguation using FICLONE NED

DBpedia Spotlight is a general semantic annotator, so the spotted mentions can contain common names, like *president* and *company*. For this reason DBpedia Spotlight obtains a low precision for datasets where only named entities are recognized. To fix this issue, we use an existent named entity recognizer, Stanford NER [14], as a tool to distinguish between a named entity and a common name. We also use this named entity recognizer to fix some spotting errors made by Spotlight and to spot relevant mentions that DBpedia Spotlight is not able to detect.

DBpedia Spotlight offers a parameter, called *confidence*, that allows the users to balance between precision and recall. Assigning the value 0 to this parameter means that we want DBpedia Spotlight to annotate every spotted mention, while with value 1 we obtain only the annotations for which DBpedia Spotlight is sure of the correctness of the linked entity. To determine which annotations will be retained, DBpedia Spotlight computes the difference between the score of the best candidate entity e_1 and the second best candidate entity e_2 for the same mention m. If their scores are close, DBpedia Spotlight will return the annotation $m \rightarrow e_1$ only if the value assigned to the confidence parameter is low. Put simply, the larger the confidence value, the greater the score difference must be in order to keep the best candidate. DBpedia Spotlight also offers a service [3] that returns all the best candidates for every mention. We use this service in our collective disambiguation step. Figure 5.1 illustrates the architecture of FICLONE NED. We will now explain each step in detail.

Spotting

To spot relevant mentions, we rely entirely on the output of the named entity recognizer. Stanford NER [4] is based on Conditional Random Fields (CRF) models and is widely used in the development of NLP applications. In an evaluation of named entity recognizers on bibliographical texts [2] and microposts [12], Stanford NER was identified as one of the best systems. We thus decided to use it to extract relevant named entities. Our main idea was to boost the performance of DBpedia Spotlight on named entities, so we used the recommended confidence value, 0.5, to annotate the input text. At this level we have two sets of mentions : M_D, the mentions that are returned by both the named entity recognizer and DBpedia Spotlight (i.e $NER \cap Spotlight$) and M_A, the mentions that are returned only by the named

3. http ://spotlight.sztaki.hu :2222/rest/candidates

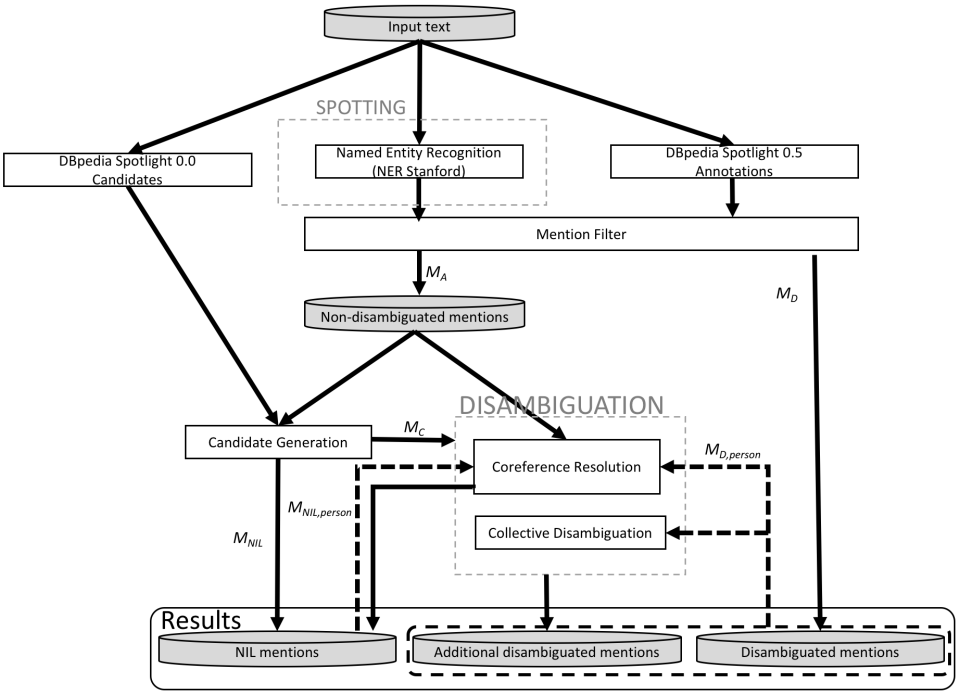


Figure 4.1 FICLONE NED architecture

entity recognizer. We ignore the mentions that are returned only by DBpedia Spotlight, since they could correspond to concepts that are not named entities. The step of separating these sets of mentions is identified in Figure 5.1 as *Mention filter*.

Candidate generation

The mentions contained in the set M_A are not linked to any entity in the knowledge base. We thus need to identify candidate entities for these mentions. For some of these entities, we obtain a list of candidates by using DBpedia Spotlight (the service that returns the best candidates for each mention) with confidence value at 0.0 (remember that in this case, we get much more mentions than at confidence 0.5). This is not sufficient : some mentions detected by Stanford NER are not recognized by DBpedia Spotlight 0.0, and the list of candidates returned by DBpedia Spotlight 0.0 does not always contain the right candidate, as shown in Figure 4.2. In this example, DBpedia Spotlight properly annotates the mention *Stefan Schumacher* with the entity *Stefan_Schumacher*, while the list returned for the mention *Schumacher* does not contain the entity *Stefan_Schumacher*. For this reason, we need another source of candidates.

In a first experiment, we ran different SPARQL requests on DBpedia to identify candidates,

Figure 4.2 Spotlight Snapshot

using the *rdfs :label* predicate, but it didn't work very well. For example, we were not able to find the right candidate *Stefan_Shumacher* for the mention *Schumacher*, because in our requests we used the exact match. Using more flexible requests was not an option because it would take too much time (a flexible request can take about 2 minutes to return results). To solve this problem, we exploit the dataset introduced in [3]. In this paper, the authors present different datasets, where each surface form is linked to a set of candidates extracted from Wikipedia. A score of TF-IDF is also indicated when the surface form comes from Wikipedia *anchors text* (i.e, the segment of text that is associated to a wikilink). We follow their recommendation to use LRD&WAT [4] surface forms filtered at TF-IDF threshold of 2.6, which is deemed to be well-suited for tasks that require high precision, as it is the case in our requirements.

Using this dataset and the candidates found by DBpedia Spotlight 0.0, we obtain a new set M_C of mentions for which we have candidates (see Figure 5.1). The remaining mentions, that is, the ones that have been detected by the NER and for which we could not find any candidate, are annotated by **NIL**, to indicate that we did not find any corresponding entry in DBpedia.

Disambiguation

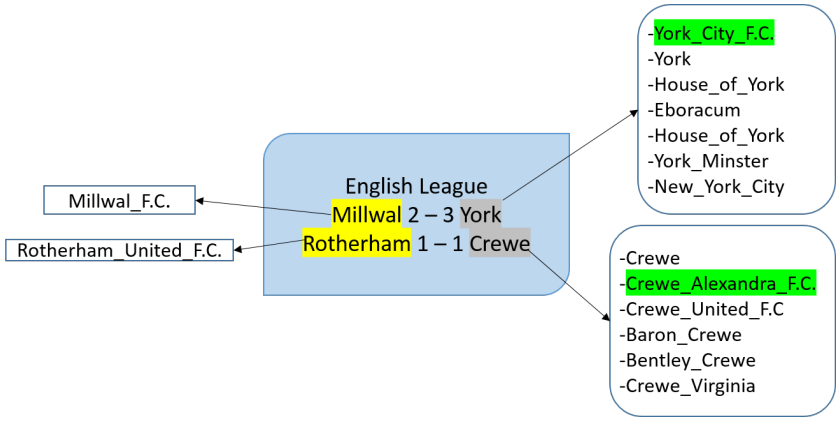
Our main focus in this experiment is to enhance the performance of DBpedia Spotlight on the named entity disambiguation task, by using a collective disambiguation process. In this method, every ambiguous mention m is disambiguated by taking into account the decisions

4. http ://data.dws.informatik.uni-mannheim.de/dbpedia/nlp2014/lrd-wat/

that have been made for other mentions $n \neq m$.

Consider for example the annotated snippet illustrated at Figure 4.3. Here we see a small table that gives the final results of two games of the English football League. As we can notice, DBpedia Spotlight correctly annotated *Millwal* with *Millwall_F.C.* and *Rotherham* with *Rotherham_United_F.C.*, but *Crewe* and *York* remain unlinked, since these mentions were not spotted by Spotlight. However, these last two mentions were detected by Stanford NER and we show, for each one, the list of candidates returned by the method described in the previous section. Now, the challenge is to select the best candidates (highlighted in green) by taking into acount the two already disambiguated mentions (*Millwal* and *Rotherham*). Here, we should be able to consider the fact that they correspond to football clubs.

Figure 4.3 DBpedia Spotlight outputs



To disambiguate the entities in M_A (the ones spotted by Stanford NER), we first use a coreference resolution process dedicated to the identification of persons, and a collective disambiguation method to disambiguate the remaining mentions.

Coreference resolution

The coreference resolution method is implemented to avoid the problem of choosing the wrong candidate in cases where its name is only partially specified in the mention, whereas the full name is used in another disambiguated mention, as in the example of *Schumacher* illustrated at Figure 4.2. With 0.5 as confidence value, DBpedia Spotlight correctly annotates the person mentions when the full name is mentionned in the text, like *Stefan Schumacher* $\rightarrow$ *Stefan_Schumacher* in our example. When the name is not fully specified (i.e *Schumacher*), instead of generating more candidates to this mention, we directly associate it with *Stefan_Schumacher*. To detect this situation we proceed as follows :

First, we define a subset P of DBpedia entities that are given one of the following types :

— http ://dbpedia.org/ontology/Person
— http ://xmlns.com/foaf/0.1/Person
— http ://schema.org/Person

We exploit different namespaces because we noted in our experiments that they were complementary to identify persons. We used the set P to implement our coreference resolution process, which is decomposed into two main steps. Firstly, we identify the subset of mentions already diambiguated by DBpedia Spotlight that correspond to persons, that is, the set $M_{D,person} \subseteq M_D$, which contains every mention $m_i \in M_D$ that is linked to an entity $e_{person} \in P$. Secondly, we extract from M_A (remember that M_A is the set of mentions spotted by Stanford NER that are not diasambiguated by DBpedia Spotlight, as shown in Figure 5.1) the mentions that are a substring of one of the mentions $m_k \in M_{D,person}$ and link them to the same entity as the one associated to m_k. Note that after this second step, a mention may still have more than one candidate. For example, let's suppose that we have three mentions, *Schumacher* $\in M_A$, *Stefan Schumacher* $\in M_D$, linked to entity e_1 and *Elizabeth Schumacher* $\in M_D$, linked to e_2. In this case we assign both candidates e_1 and e_2 to the mention *Schumacher*.

We implemented a similar approach to deal with mentions annotated with NIL (set M_{NIL} in Figure 5.1). The NER used in FICLONE NED does not only spot named entities, it also assigns them a type (Person, Organization, Location and MISC). We use this typing to identify the subset $M_{NIL,person} \in M_{NIL}$, the set of mentions that the NER types as a PERSON. We apply the same process described above to detect every mention $m \in M_C$ that is a substring of a mention in $M_{NIL,person}$. Since in this case there is no candidate entity, we link the mention m to NIL, and thus avoid an incorrect disambiguation with another entity that would exist in DBpedia. For example, the mention *Majed Shehadeh*$\in M_{NIL}$ has no entries in DBpedia, while the mention *Shehadeh*$\in M_C$ has several candidates, so when these two mentions appear in the same text, we annotate both of them with **NIL**.

Collective disambiguation

The collective disambiguation is applied to the set M_C, that is, the set of mentions detected by Stanford NER for which we could generate a list of candidate entities. We use in our approach two metrics : a *direct score* and a *coherence score*. The direct score corresponds to the number of times a candidate e of a mention $m \in M_C$ is linked to the entities assigned to the mentions M_D (the ones that have been disambiguated by DBpedia Spotlight). Note that after disambiguating a mention $m \in M_C$, this mention is added to the set M_D and will be

used to annotate other mentions from the set M_C. In our experiments we encountered cases where some entities have the same direct score. The coherence score is used to discriminate between these entities.

Our direct score, which is assigned to each candidate e_c of a mention m, is inspired from the one used in Semlinker [7], an annotator that uses collective disambiguation. This score is based on the corresponding Wikipedia links of the entity e_c and is defined as follows :

$$DirectScore(e_c) = log(\frac{card(\{e_i | e_i \in M_D \text{ and } e_c \in links(e_i)\}}{card(M_D)} + 1) \tag{4.1}$$

where e_c is the candidate entity, M_D is the set of annotations that have already been disambiguated, and $links(e_i)$ is the set of links in which e_i is involved as source (outlinks) or destination (inlinks). The direct score reflects how many times a candidate e_c appears among the links of the entities that are already disambiguated. We experimented with both inlinks and outlinks. We do not report in this article all the results, but in our experiments we obtained better performances with outlinks. It seems that the occurrences of an entity e_c in the context of the entities that are already disambiguated is more relevant than the frequency of disambiguated entities in the context of e_c. We thus use only outlinks in the direct score.

Using the direct score metric is not always enough to discriminate between the entities : we observed that in some cases a set of candidates associated to the same mention have the same direct score value. As an example, for the mention "U.S." we obtained the same score for entities *United_States*, *United_States_dollar* and *United_States_Armed_Forces*. In some other cases, all entities have a direct score of 0. Another score was needed to compute the coherence of e_c with M_D. The coherence score $Coh(e_c)$ expresses how much a candidate entity is semantically related to the other entities already disambiguated. Supposing that $Sim(e_c, e_d)$ represents the relatedness of candidate e_c with an already disambiguated entity $e_d \in M_D$, the coherence score is computed by averaging over the values obtained for all entities in M_D :

$$Coh(e_c) = \frac{1}{|M_d|} \sum_{e_d \in M_D} Sim(e_c, e_d) \tag{4.2}$$

To calculate $Sim(e_c, e_d)$, we considered two well-known formulas : the relatedness metric introduced in [38] and the Jaccard similarity measure [18]. The coherence score is computed only for the candidates that share the best direct score, the rest of candidates being ignored :

$$relatedness(e_a, e_b) = \frac{log(max(|A|, |B|)) - log(A \cap B)}{log(N) - log(min(|A|, |B|))} \tag{4.3}$$

$$Jaccard(e_a, e_b) = \frac{(|A \cap B|)}{(|A \cup B|)} \tag{4.4}$$

Where e_a and e_b are two entities of interest, A and B are the set of entities that are respectively linked to a and b, while N is the total number of entities in Wikipedia. Note that for computing the coherence score, we can consider the shared inlinks or the shared outlinks to determine whether two entities are linked.

Jaccard similarity computed with outlinks is used in FICLONE, since it performs slightly better than relatedness, according to our experiments, whose results are presented in Section 4.4.2.

4.3.2 Semantic annotation using FICLONE SA

In this section we describe our approach to improve DBpedia Spotlight on semantic annotation, where named entites as well as other concepts are annotated. The architecture of FICLONE SA is represented in Figure 4.4. It differs from FICLONE NED on the following aspects (indicated in red in the figure) : it is not limited to the mentions spotted by Stanford NER and the mention filter is replaced by a two-step process (selection plus filtering). Also, since the main purpose of the semantic annotation is to link the spotted mentions to an existing knowledge base, NIL mentions are not annotated in FICLONE SA. We will now explain these differences in details.

Contrarily to FICLONE NED (Figure 5.1), which uses only the output of Stanford NER, FICLONE SA also exploits the mentions detected by DBpedia Spotlight at confidence 0, thus maximizing the number of detected mentions. In our experiments we noticed that some mentions from the two sources can overlap. For example, DBpedia Spotlight may detect the mention "England football team", where Stanford NER would return only "England". In these cases, we always keep the longest mention (mention selection). We also noticed that many mentions returned by DBpedia Spotlight at confidence 0 represent only adverbs, adjectives or pronouns. To fix this issue, we filter out these mentions using the Stanford POS tagger. As in FICLONE NED, the mentions disambiguated by DBpedia Spotlight 0.5 are directly added to the results. But, differently from FICLONE NED, the mentions that are not detected by Stanford NER are also added to the results (set C_D). They correspond to concepts that are not named entities. For this reason, they will not be used in the collective disambiguation process, since the mentions to be disambiguated all refer to named entities found by Stanford NER.

Thus, at the end of the spotting phase, we obtain three mentions sets : M_D, the mentions that

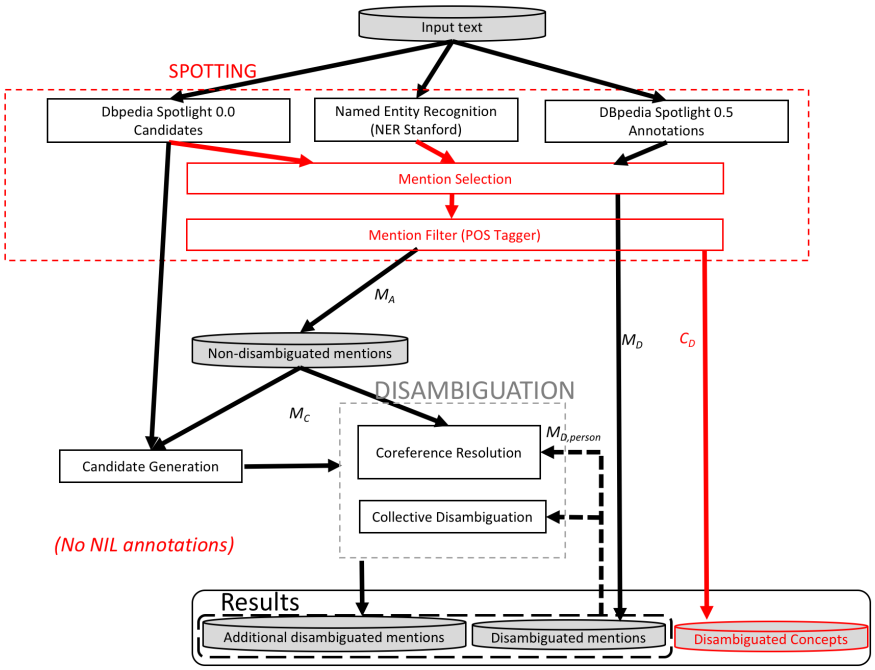


Figure 4.4 FICLONE SA architecture

are returned by DBpedia Spotlight 0.5 and Stanford NER ; M_A, the mentions returned only by Stanford NER and a new set C_D, which contains the spots returned by DBpedia Spotlight 0.5 that are not detected by Stanford NER. For the disambiguation step, FICLONE SA uses the same techniques as in FICLONE NED for the set M_A, namely the coreference resolution as well as the collective disambiguation process. Note that annotations in sets C_D and M_D are returned directly without further disambiguation.

4.4 Evaluation methodology

4.4.1 Experimental setup

In the previous section, we described two annotation services based on DBpedia Spotlight : FICLONE NED, which transforms it into a genuine name entity disambiguator, and FICLONE SA, which essentially complements it with a named entity recognizer and a collective disambiguation process.

To evaluate these two systems, we used two different kinds of datasets : NED datasets, where all the named entities are identified and SA datasets, where named entities as well as concepts are identified.

NED datasets :

— AIDA-ConLL : it is a collection of 3393 news texts from *Reuters news stories* manually annotated by AIDA developers.
— KORE 50 : this dataset is extracted from AIDA-ConLL. It contains 50 short sentences with ambiguous named entities.
— MSNBC : this dataset, introduced in [10], contains 20 news stories extracted from MSNBC. Note that only the most relevant named entities are annotated.
— N3 Reuters 128 : this dataset was introduced in [32]. It contains 128 economic news aticles extracted from *Reuters news stories*.
— N3 RSS 500 : this dataset was introduced in [32]. The authors extracted RSS feeds from major worldwide newspapers. Then they manually annotated 500 sentences randomly chosen.

SA datasets :

— DBpedia Spotlight dataset [5] (Spotlight DS) : this dataset is composed of 58 sentences extracted from the New York Times.
— Tagme dataset [6] (Tagme DS) : this dataset is composed of 180,000 short snippets of text extracted from Wikipedia 2009. In this dataset, the mentions are annotated with the Wikipedia ID page. Note that some of these IDs are not valid in the current version of Wikipedia. We thus removed from the dataset the sentences that contain a mention associated to an ID page that does not exist anymore. This reduced the dataset to 172,473 sentences. From these sentences we chose the first 10000 ones and used them for the evaluation of our system. This reduced dataset contains 47554 annotations.

Table 5.1 presents some descriptive statistics about these datasets. As we can notice, all the datasets contain NIL annotations for mentions that do not have any corresponding entries in the target knowledge base, except for KORE 50, Spotlight DS and Tagme DS, where NIL mentions are ignored.

To compare the performances of FICLONE, we used the following five state-of-the-art systems :

— DBpedia Spotlight : we used the default configuation (i.e confidence = 0.5)
— AIDA : AIDA returns every mentions that Stanford NER is able to spot. We noticed that for some mentions, AIDA leaves the "Kbidentifier" field empty. We considered these mentions as NIL mentions.

5. yovisto.com/labs/ner-benchmarks/
6. acube.di.unipi.it/tagme-dataset/

Table 4.1 Statistics on the datasets

Type	# documents	# mentions	# disamb. mentions	# NIL mentions
AIDA-ConLL	1393	34929	27817	7112
KORE 50	50	144	144	0
MSNBC	20	747	654	93
N3 reuters 128	128	880	631	249
N3 RSS 500	500	1000	522	478
Spotlight DS	10	329	329	0
Tagme DS	10000	47554	47554	0

— Tagme : we used the default configuration.
— Babelfy : we used the "NAMED_ENTITIES" option.
— WAT : we used the default configuration.

To evaluate the performances of each system we calculated precision and recall as well as the F1-score as a final score. NIL annnotations are not considered in the computation of these scores. We also conducted three types of experiments :

— A2KB : an annotation is counted as a true positive only when the mention has been both correctly spotted and disambiguated.
— Entity Spotting (ES) : an annotation is counted as a true positive only when the mention has been correctly spotted.
— Entity Disambiguation (ED) : an annotation is counted as a true positive only when the mention has been rightly disambiguated. Note that ED is different from A2KB, since the mentions that have been wrongly spotted are not considered in this evaluation.

4.4.2 Selection of metric for coherence score

As described in Section 4.3.1, the direct score metric is not always sufficient to discriminate between the candidate entities, because sometimes all the top-ranked candidates have the same direct score. Thus, to determine the appropriate additional metric (relatedness or Jaccard) that should be used to compute the coherence score, we tested both on our datasets. We also experimented with the two following configurations : using inlinks and using outlinks. The F-Score of each formula is given in Table 4.2.

We notice that the Jaccard metric produces better results on both long (AIDA-ConLL , MSNBC and N3 Reuters 128) and short (KORE50 and N3 RSS 500) texts. Additionnally, computing the Jaccard distance with the outlinks gives better results in 3 datasets among 5. We thus adopted the Jaccard metric with outlinks for the computation of the coherence score. Another incentive to use outlinks is their lower number of occurrences, which makes it faster to compute the coherence metric (i.e, the entity *Canada* has 124410 inlinks and 622 outlinks).

Table 4.2 Comparison of F-scores obtained with Jaccard and relatedness metrics for disambiguation

	Jaccard		relatedness	
	Inlinks	Outlinks	Inlinks	Outlinks
AIDA-ConLL	74.02	75.82	74.33	74.84
KORE50	49.81	46.79	46.79	45.28
MSNBC	75.2	76.39	75.06	75.8
N3 Reuters 128	59.08	59.46	58.71	59.08
N3 RSS 500	67.64	67.28	67.06	66.77

4.4.3 Performance of state-of-the-art annotators

Before presenting the improvement obtained by coupling DBpedia Spotlight with the methods implemented in FICLONE NED and FICLONE SA, it is important to see how DBpedia Spotlight compares to other state-of-the-art annotators. Table 4.3 reports the results for the A2KB subtask. We highlight in red the best system and in blue the second best system.

We can notice that the two semantic annotators (i.e DBpedia Spotlight and Tagme) dominates on Spotlight DS and Tagme DS, while the other ones perform better on the NED datasets (note that Babelfy's performances are not very good, compared to WAT and AIDA). This confirms that semantic annotators and named entity disambiguators must be evaluated separately. It may be strange to see WAT classified as a NED system, since it is an improvement of Tagme, which is a SA system, but as we can see in Table 4.3, WAT's results clearly show that it is a named entity disambiguator (second best on NED datasets and poor performances on SA datasets) [7]. So in our evaluation, we will present the results of FICLONE NED on the NED datasets and compare it to NED systems (AIDA, WAT and Babelfy) while for FICLONE SA, we will test it on the SA datasets and compare it with SA systems (DBpedia Spotlight and Tagme).

7. In fact, by a manual inspection of the WAT's results, we noticed that only named entities are annotated.

Table 4.3 Performances of NED and SA systems

	SA						NED								
	Spotlight			Tagme			WAT			AIDA			Babelfy		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AIDA-ConLL	0.52	0.49	0.51	0.19	0.45	0.27	0.62	0.61	0.61	0.72	0.7	0.71	0.31	0.46	0.37
KORE 50	0.37	0.22	0.28	0.3	0.54	0.39	0.48	0.43	0.45	0.66	0.52	0.58	0.53	0.55	0.54
MSNBC	0.42	0.43	0.43	0.11	0.54	0.18	0.54	0.5	0.52	0.67	0.6	0.63	0.26	0.53	0.35
N3 Reuters 128	0.19	0.26	0.22	0.05	0.3	0.09	0.29	0.36	0.32	0.45	0.52	0.48	0.13	0.28	0.18
N3 RSS 500	0.23	0.31	0.26	0.08	0.36	0.13	0.2	0.33	0.25	0.43	0.62	0.51	0.12	0.31	0.17
Spotlight DS	0.54	0.24	0.34	0.3	0.6	0.4	0.27	0.11	0.15	0.3	0.11	0.16	0.12	0.08	0.1
Tagme DS	0.62	0.57	0.59	0.36	0.72	0.48	0.46	0.36	0.4	0.45	0.35	0.39	0.29	0.45	0.36

Now considering DBpedia Spotlight, we see that its performances on NED datasets are always lower than the two best SA systems (WAT and AIDA), with the exception of N3 RSS 500, where it slightly outperforms WAT. Interestingly, for the SA task, DBpedia Spotlight obtains the best results on Tagme DS, while the situation is exactly the opposite on Spotlight DS. Looking more closely at the results of these two annotators on SA datasets, we observe that DBpedia Spotlight's weakness is its recall, which is exactly what we expect to improve with the methods implemented in FICLONE SA.

4.4.4 Evaluation of FICLONE NED

In this section, we present the result of several experiments. First, we show that Stanford NER greatly improves the spotting step. Second, we evaluate the performance of our collective disambiguation approach. Third, we consider the full task and compare FICLONE NED with the other state-of-the-art NED systems.

The impact of using Stanford NER for spotting

Stanford NER is a named entity recognizer : it is able to spot relevant named entities and do not disambiguate them. Thus its impact is only on the spotting step (ES subtask). Table 4.4 shows the performances of DBpedia Spotlight and Stanford NER on all the selected datasets. It demonstrates the advantage of using Stanford NER in our implementation. As noted earlier, DBpedia Spotlight does not spot only named entities, which explains its low precision.

Since Stanford was designed to spot specifically named entities, it is not surprising to observe that it performs much better than DBpedia Spotlight in all the datasets. We analyzed the outputs of DBpedia Spotlight to understand this huge difference in performance. Figure 4.5 highlights the main types of errors we found.

— Problem with the dot "." : Spotlight recognizes *F.A.*, while it should be *F.A. cup*, but

Table 4.4 Performances of Stanford NER and DBpedia Spotlight for the entity spotting substask

	Stanford NER			Spotlight		
	P	R	F	P	R	F
AIDA-ConLL	0.98	0.97	0.97	0.67	0.63	0.65
KORE 50	0.95	0.87	0.91	0.62	0.37	0.46
MSNBC	0.77	0.79	0.78	0.48	0.49	0.49
N3 Reuters 128	0.68	0.81	0.74	0.25	0.32	0.28
N3 RSS 500	0.58	0.86	0.69	0.33	0.42	0.37

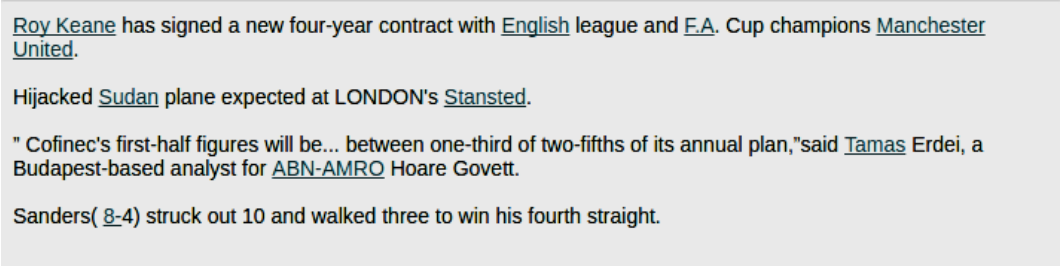


Figure 4.5 DBpedia Spotlight errors

when we replace the point with a space " ", Spotlight recognizes the entire mentions *F A Cup*.

— Spotting in Spotlight is case sensitive. Spotlight was able to recognize "LONDON" only when we changed it to "London". We assume that the surface form "LONDON" has never been met when Wikipedia was processed to build the surface form lexicon.

— The fact that DBpedia Spotlight does not consider the mentions that do not exist in DBpedia causes many errors. For example, since *Tamas Erdei* has no entry in DBpedia, Spotlight not only ignores this mention but also mistakenly spots "Tamas" and annotates it with "dbpedia.org/resource/Tamas_(philosophy)".

— The identification of numbers. For example, Spotlight may annotate the mention *8-* with entity dbpedia/resource/8-bit", which has no relation with the context of the input text.

Table 4.5 compares the performances of FICLONE NED for spotting (which are in fact the same as Stanford NER) to the other NED systems. We can notice that the results of FICLONE NED and AIDA for the entity spotting task are the best on all the datasets. These two systems both use Stanford NER, which supports our choice in using it as the main component for the spotting phase. The minor differences could be explained by the fact that FICLONE NED uses the latest version of Stanford NER while AIDA uses an oldest one.

Table 4.5 Comparison with other NED systems for spotting (ES)

	FICLONE NED			AIDA			Babelfy			WAT		
	P	R	F	P	R	F	P	R	F	P	R	F
AIDA-ConLL	0.98	0.97	0.97	0.97	0.94	0.96	0.45	0.65	0.53	0.82	0.81	0.82
KORE 50	0.96	0.87	0.91	0.94	0.77	0.84	0.67	0.69	0.68	0.87	0.74	0.80
MSNBC	0.77	0.79	0.78	0.84	0.75	0.79	0.33	0.66	0.44	0.70	0.65	0.67
N3 Reuters 128	0.68	0.81	0.74	0.74	0.82	0.78	0.19	0.40	0.26	0.37	0.47	0.42
N3 RSS 500	0.58	0.86	0.69	0.60	0.84	0.70	0.21	0.47	0.29	0.35	0.53	0.42

The impact of collective disambiguation

Tables 4.6 provides a comparison of systems' performances for the disambiguation subtask. These results are obtained by taking every correctly spotted mention and evaluating the correctness of the entity that has been linked to this mention. We can see that FICLONE NED not only substantially improves the results of DBpedia Spotlight disambiguation, but also outperforms the other state-of-the-art annotators for all datasets, except KORE 50. We can also notice that FICLONE NED has the best precision for the entity disambiguation task (ED) in all the datasets except MSNBC, where DBpedia Spotlight is slightly better (0.89 vs 0.87). As expected, the good performances of FICLONE NED are mainly due to the fact that it obtains a better recall than DBpedia Spotlight. From these results, we can conclude that FICLONE NED is more efficient in detecting the right candidate when the mention is correctly spotted.

Comparison to state-of-the-art systems for full task

We compared the performances of FICLONE NED for the full task with the state-of-the-art annotators. The results are shown in Table 4.7. We can observe that FICLONE NED is more competitive than DBpedia Spotlight, when compared with the other systems. The performances of FICLONE NED are the best on MSNBC and AIDA-ConLL corpora, while AIDA is slightly better on N3 Reuters 128 and N3 Reuters 500. AIDA outperforms all the annotators on KORE 50. This dataset contains short sentences with very ambiguous mentions, thus making the task of semantic annotation very difficult. As discussed in Section4.5.2, the low performance of FICLONE on this dataset, compared to other systems, is due to the wrong annotations made by DBpedia Spotlight that are directly transmitted to FICLONE's output.

4.4.5 Evaluation of FICLONE SA

To evaluate the perfomance of FICLONE SA, we compare it to Tagme and DBpedia Spotlight 0.5 on the Spotlight DS and Tagme DS datasets. The results are reported in Tables 4.8, 4.9 and 4.10.

Table 4.6 Comparison with other NED systems for disambiguation (ED)

	FICLONE NED			Spotlight			AIDA			Babelfy			WAT		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AIDA-ConLL	0.77	0.75	0.76	0.78	0.49	0.60	0.74	0.70	0.72	0.69	0.46	0.55	0.75	0.61	0.67
KORE 50	0.48	0.43	0.45	0.40	0.22	0.28	0.68	0.52	0.59	0.69	0.55	0.61	0.52	0.43	0.47
MSNBC	0.87	0.68	0.77	0.89	0.43	0.58	0.80	0.60	0.69	0.79	0.53	0.63	0.77	0.50	0.60
N3 Reuters 128	0.61	0.53	0.57	0.53	0.26	0.35	0.61	0.52	0.56	0.51	0.28	0.36	0.59	0.36	0.45
N3 RSS 500	0.72	0.63	0.67	0.44	0.31	0.36	0.71	0.62	0.66	0.44	0.31	0.36	0.44	0.33	0.38

Table 4.7 Comparison with other NED systems for the full process (A2KB)

	FICLONE NED			Spotlight			AIDA			Babelfy			WAT		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AIDA-ConLL	0.75	0.75	0.75	0.52	0.49	0.51	0.72	0.70	0.71	0.31	0.46	0.37	0.62	0.61	0.61
KORE 50	0.46	0.43	0.44	0.37	0.22	0.28	0.66	0.52	0.58	0.53	0.55	0.54	0.48	0.43	0.45
MSNBC	0.67	0.68	0.68	0.42	0.43	0.43	0.68	0.6	0.64	0.26	0.53	0.35	0.54	0.5	0.52
N3 Reuters 128	0.43	0.53	0.47	0.19	0.26	0.22	0.45	0.52	0.48	0.13	0.28	0.18	0.29	0.36	0.32
N3 RSS 500	0.41	0.63	0.5	0.23	0.31	0.26	0.43	0.62	0.51	0.12	0.31	0.17	0.2	0.33	0.25

Table 4.8 Comparison with other SAs for the full process (A2KB)

	FICLONE SA			Spotlight			Tagme		
	P	R	F	P	R	F	P	R	F
Spotlight DS	0.48	0.56	0.52	0.54	0.24	0.34	0.30	0.60	0.40
Tagme DS	0.41	0.65	0.50	0.62	0.57	0.59	0.36	0.72	0.48

Table 4.9 Comparison with other SAs for spotting (ES)

	FICLONE SA			Spotlight			Tagme		
	P	R	F	P	R	F	P	R	F
Spotlight DS	0.59	0.65	0.62	0.61	0.27	0.37	0.4	0.82	0.53
Tagme DS	0.47	0.74	0.57	0.70	0.63	0.66	0.43	0.85	0.57

Table 4.10 Comparison with other SAs for disambiguation (ED)

	FICLONE SA			Spotlight			Tagme		
	P	R	F	P	R	F	P	R	F
Spotlight DS	0.76	0.56	0.65	0.65	0.24	0.35	0.73	0.60	0.66
Tagme DS	0.86	0.65	0.74	0.85	0.57	0.68	0.84	0.72	0.78

From Table 4.8, we notice that FICLONE SA obtains the best results against the DBpedia Spotlight dataset. This is mainly due to recall, which increases substantially (from 0.34 to 0.52) for the whole annotation process. Precision decreases (from 0.54 to 0.48) mainly due to spotting, which is noisier, as shown in Table 4.9 (precision of 0.59 instead of 0.61). Both precision and recall are improved on the disambiguation step (see Table 4.10). Now comparing to Tagme's performances on the same dataset, we also observe that FICLONE SA performs better for the full task (F-Score of 0.52 for FICLONE, vs 0.40 for Tagme, according to Table 4.8), mainly due to a much better precision (0.48 for FICLONE, vs 0.30 for Tagme). From the results on this dataset, we can conclude that the main problem of DBpedia Spotlight is its performance on recall, which is exactly the aspect that is improved in FICLONE SA.

For the Tagme dataset, our analysis is different. Recall is improved, compared to DBpedia Spotlight, but the degradation on precision is worse than on the other dataset. It seems that the decrease in precison for spotting we can observe in Table 4.9 is not compensated by the increase in precision for disambiguation (see Table 4.10). This phenomemon can be explained by the fact that, contrarily to Spotlight DS, not all relevant mentions have been marked in Tagme DS, thus penalizing the semantic annotator. For instance, let's consider the following text fragment :

> *. . . is found in caves through Kentucky and southern Indiana. It is listed as a threatened species in the United States and the IUCN lists the species as vulnerable. . . .*

Mentions *United States* and *caves*, which are correctly annotated by FICLONE SA, are not marked in Tagme DS. For this reason, it is expected to observe a decrease in precision for the spotting subtask. If we consider only the disambiguation subtask (Table 4.10), we see that the performance is improved compared to DBpedia Spotlight, as it was the case with Spotlight DS (from 0.68 to 0.74 for F-score). This indicates the good potential of the collective disambiguation process implemented in FICLONE SA. Tagme has the best recall and F-score for entity disambiguation on this dataset, but FICLONE SA obtained better performances than DBpedia on these metrics. In fact, these performances are closer to the results obtained by Tagme.

4.5 Analysis of results

In this section we analyze the outputs of FICLONE against AIDA-ConLL and MSNBC to explain its errors for the spotting and disambiguation steps.

4.5.1 Analysis of spotting results

To show the importance of using Stanford NER for spotting, Table 4.11 provides some statistics on the performances of three spotting approaches on AIDA-ConLL and MSNBC datasets : the one used in DBpedia Spotlight 0.5, the one used in FICLONE NED (which in fact is the same as Stanford NER), and $DS \cap ST$, which is the intersection of the mentions returned by NER Stanford with the mentions returned by Dbpedia Spotlight 0.5. For each, we give the total number of spotted mentions and the number of correct and incorrect spots.

Table 4.11 shows the problem of DBpedia Spotlight in filtering irrelevant mentions. On AIDA-ConLL dataset, DBpedia Spotlight generates 9007 wrong mentions, out of 31758, which represents 28% of the total, while using Stanford to filter the output of Spotlight decreases the number of wrong mentions from 9007 to 131, at the cost of losing 313 right mentions. The same case can be observed on MSNBC, where the number of irrelevant mentions decreases from 385 to 64 mentions. But using the intersection between Stanford and DBpedia Spotlight only increases the precision of DBpedia Spotlight, while using only the output of Stanford NER (the solution used in FICLONE NED) greatly increases the recall : from 22751 to 34062 correct spots on AIDA-ConLL and from 353 to 604 on MSNBC.

Stanford NER errors

As shown in Section 4.4.4, Stanford NER obtains a very high F-Score for AIDA-ConLL dataset, while it generates a greater ratio of errors for MSNBC. Since MSNBC does not annotate all the occurrences of relevant mentions as well as the modifiers like *American, German* that are marked on the AIDA-ConLL dataset (which causes 56 wrong spots), we focus only on cases where a mention from Stanford NER overlaps a mention in the Gold Standard. We found two kinds of errors :

— Mentions that should be separated (example : *Highmark Blue Cross Blue Shield of Western Pennsylvania* that should be separated into *Highmark Blue Cross Blue Shield* and *Western Pennsylvania*). We noted this kind of errors 19 times on MSNBC.

Table 4.11 Impact on spotting

Dataset	Spotlight(0.5)			$DS \cap ST$			FICLONE NED		
	#spots	#correct	#wrong	#spots	#correct	#wrong	#spots	#correct	#wrong
AIDA-ConLL (34929 mentions)	31758	22751	9007	22569	22438	131	34672	34062	610
MSNBC (747 mentions)	738	353	385	368	304	64	761	604	157

— Mentions that should be enlarged instead, (example : *University of Alabama* that should be *University of Alabama at Birmingham*). These errors occur 54 times on MSNBC.

These errors could be fixed by some heuristic-based technique that recognizes the composition of complex nominal phrases, such as the one proposed in [25].

4.5.2 Analysing the disambiguation step

To analyze the errors made by FICLONE in linking ambiguous mentions, once again we focus on AIDA-ConLL and MSNBC datasets. In FICLONE, there are three sources of annotations : DBpedia Spotlight 0.5, the coreference resolution and the coherence measure based on direct score and Jaccard. The number of disambiguations returned by each one of these methods are presented in Table 4.12, together with the number of correct and wrong disambiguations. Note that we disregarded the NIL annotations from the gold standard as well as from the output of FICLONE.

The coreference resolution module that we presented in Section 4.3.1 produced 1467 and 110 annotations with AIDA-ConLL and MSNBC datasets, respectively, which represents 5% and 21% of the annotations. The coherence measure is the source of 4790 (18%) and 114 (22%) annotations, respectively. Together, these two methods helped to disambiguate 23% of mentions in AIDA-ConLL dataset, and 43% in MSNBC, which is not negligible.

For AIDA-ConLL, FICLONE made 4493 errors out of 27107, but we notice that 3366 of these errors come from the DBpedia Spotlight 0.5. Only 1127 of the errors are due to the coreference resolution and coherence measure (this represents 25% of the total). Against MSNBC, we noticed that 45 errors are made by DBpedia Spotlight, while our algorithms produced 31 errors (41% of the total). This makes us conclude that to improve the performance of FICLONE, the priority should be to find a way of correcting the annotations returned by DBpedia Spotlight.

Table 4.12 Entity disambiguation analysis

Dataset	DBpedia Spotlight 0.5			Coref Resolution			Coherence			Total		
	#dis	#corr	#wr	#dis	#corr	#wr	#dis	#corr	#wr	#dis	#corr	#wr
AIDA-ConLL (27817 disamb.)	20850	17484	3366	1467	1443	24	4790	3687	1103	27107	22614	4493
MSNBC (654 disamb.)	301	256	45	110	108	2	114	85	29	525	449	76

Errors of FICLONE

We noticed three kinds of problems with FICLONE, which made it less competitive in some datasets. First, it relies completely on DBpedia Spotlight to annotate short texts. Remember that the annotations returned by DBpedia Spotight are used directly and participate to the collective disambiguation process. Thus, wrong annotations made by DBpedia Spotlight will mislead this process. See for example Figure 4.6. We highlight in red, yellow and green the outputs of DBpedia Spotlight, FICLONE (the ones found by the collective disambiguation process) and the gold standard, respectively.

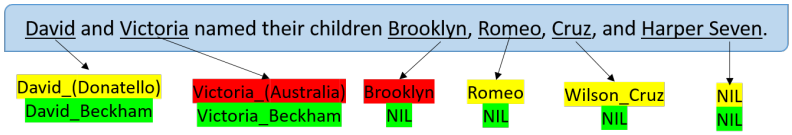


Figure 4.6 FICLONE errors

DBpedia Spotlight wrongly annotates *Victoria* with *Victoria_(Australia)*, *Brooklyn* with *Brooklyn* (the borough of New York City) In this case, it will be impossible for FICLONE to correctly link *David* to *David_Beckam*.

Second, FICLONE tries to annotate every mention for which it is able to extract candidates. In our example at Figure 4.6, we can observe that it associates *Cruz* to *Wilson_Cruz* and *Romeo* with *Romeo*, where in these cases, it should not link them to any entity. To avoid this problem, we should set a threshold in the coherence measure to filter out the low-score candidates.

Finally, in our generation of the candidate list, many possible lecxicalisations are missed. For example, the mention *Tom Moody* could also be found in an abbreviated form like *T. Moody*, which will not be part of the list of candidates. We would need to implement some rules to take into consideration these variations.

The difficulty of the disambiguation task

The semantic annotators are usually trained on some datasets, which are not the same for all these systems. For example, in AIDA-ConLL dataset, nationalities are annotated, (*French* is annotated with *http ://dbpedia.org/resource/France*), while these mentions are ignored in MSNBC. On MSNBC, the mention *President Barack Obama* is annotated as a single mention, while in AIDA-ConLL only *Barack Obama* is annotated. These differences clearly show that there is still some confusion on the task of semantic annotation, and this does not

facilitate the development of an annotator. How can we confidently evaluate our annotator if the available datasets do not agree on what should be annotated ?

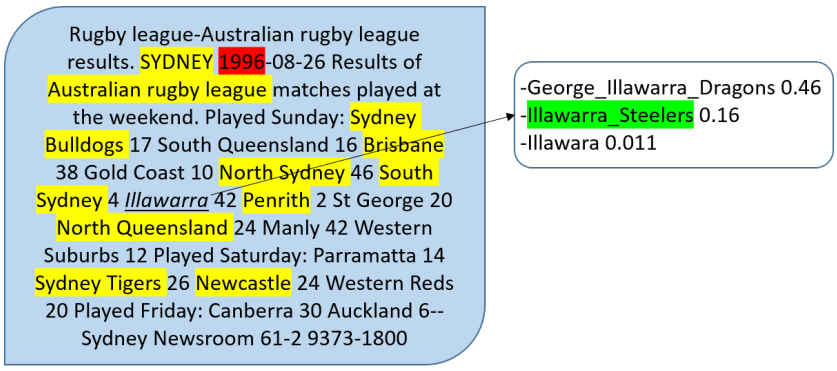


Figure 4.7 The difficulty of Entity Linking

Another important difficulty is illustrated in Figure 4.7. This example was extracted from the AIDA-ConLL dataset. The mentions that have been already disambiguated by the semantic annotator are marked in yellow. We show the list of candidates with their score for the mention *Illawarra*. In its candidate list, we see that *St._George_Illawarra_Dragons* has the highest score (0.46), while the correct entity is *Illarawa_Steelers* (0.16). Here, we clearly see the limit of collective disambiguation : all mentions already disambiguated are related to the Australian rugby league. The two best candidates are both Australian rugby teams. We cannot really expect to receive much help from collective disambiguation in this case. Some other kind of inference must be used to disambiguate this mention. Here, for example, according to the information we can obtain regarding *St._George_Illawarra_Dragons*, this club was founded in 1998, while the results reported in the input text date from 1996, so it cannot be the correct entity.

4.6 Conclusion

In this paper, we have shown that using a named entity recognizer for the spotting module, and a collective approach for disambiguation, substantially improves the performance of DBpedia Spolight. For the identification of candidates URIs that correspond to some mention in the text, we used DBpedia Spotlight *Candidates* service at confidence 0.0, combined with an external source of candidates. For the collective disambiguation process, we introduced a direct score based on the outlinks of each wikipedia candidate, combined with the Jaccard score used to compare each candidate to other disambiguated entities. We also demonstrated the positive impact of the coreference resolution to boost the performance of the disambiguation process. Our experiments not only showed our success in enhancing the performances of

DBpedia Spotlight for the NED task, but also that the resulting system, FICLONE NED, outperforms the best semantic annotators publicly available in 4 out of 5 datasets. We also showed that by combining the outputs of DBpedia Spotlight and Stanford NER, which resulted in FICLONE SA, we also boosted the performances of DBpedia Spotlight.

For future work, we first plan to introduce some linguistic methods to fix the errors made by Stanford NER. Secondly, some mechanism should be implemented to correct the annotations made by DBpedia Spotlight before using them in our disambiguation module. We should also implement a dynamic candidate generator that would take into account the context given by the text. We also plan to implement a machine learning method to enhance the performance of FICLONE with short texts, and make it less dependent of DBpedia Spotlight.

4.7 Aknowledgement

This research has been funded by the NSERC Discovery Grant Program.

CHAPITRE 5 ARTICLE 2 : COLLECTIVE DISAMBIGUATION AND SEMANTIC ANNOTATION FOR ENTITY LINKING AND TYPING

M. Chabchoub, M. Gagnon, and A. Zouaq. Semantic Web Challenges : Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers, chapter Collective Disambiguation and Semantic Annotation for Entity Linking and Typing, pages 33–47. Springer International Publishing, Cham, 2016.

Les coauteurs de cet article [6] sont mon directeur de recherche Mr. Michel Gagnon et ma codirectrice de recherche Mme. Amal Zouaq. Mes contributions dans cet article sont l'implémentation du système, les différentes expérimentations et la rédaction. Suite à notre succès dans la compétition *Open Knowledge Extraction*, cet article a été publié dans le livre *Semantic Web Challenges : Third SemWebEval Challenge at ESWC 2016*.

Abstract

In this paper we present the WESTLAB system, the winner of the 2016 OKE challenge Task 1. Our approach combines the output of a semantic annotator with the output of a named entity recognizer, and applies some heuristics for merging and filtering the detected mentions. The approach also applies a collective disambiguation method that relies on all the previously linked entities to choose between multiple candidate entities for a given mention. Using this approach, we greatly improve the performance of all the semantic annotators that are used as baselines in our experiments and also outperform the best system of the OKE Challenge 2015.

5.1 Introduction

The first task of the Open Knowledge Extraction (OKE) challenge is divided into three subtasks : entity recognition, entity linking and entity typing with four DUL classes (Person, Organization, Place and Role). In this paper, we show that by combining the outputs of a semantic annotator and a named entity recognizer, we can obtain very good performance for all three subtasks [1]. WESTLAB, our system, relies on the principle of collective disambiguation [23, 17], where the selection of the entity to be linked to a mention takes into account the entities previously associated to other mentions in the text. To implement our approach,

1. Our service is available at the following URL :
`http://westlab.polymtl.ca/OkeTask1/rest/annotate/post`

we propose a pipeline architecture. In the first step, the detected mentions in the text are adjusted and filtered, such that only the most relevant ones are transmitted to the entity linking module. The linking module identifies the corresponding entity in DBpedia for each mention and applies the collective disambiguation process if there are several candidates. Finally, for entity typing, our approach uses some manually defined mapping rules.

This paper is organized as follows. Next section discusses related works. Section 5.3 describes our system in more details. In Section 5.4 we present the results of our system on the training and test datasets provided by the 2016 OKE Challenge and we compare them with several baselines, using different configurations. In Section 5.5 we conclude and make some suggestions to improve WESTLAB's performance.

5.2 Related work

In this section we present some state-of-the-art approaches that perform well for named entity recognition and semantic annotation, with a particular focus on systems used in our evaluation for comparison purposes. We also present the best performing systems in the 2015 OKE challenge.

5.2.1 Named entity recognition

Extracting named entities has been tackled by numerous Natural Language Processing studies in the last decade. OpenNLP [2] uses machine learning and maximum entropy models. LingPipe [3] uses n-gram character language models. OpenCalais [4] is a commercial service that uses machine learning techniques to recognize named entities and uses a proprietary taxonomy to type them. Stanford NER [14] is based on Conditional Random Fields (CRF) models and is widely used in the development of NLP applications. In an evaluation of named entity recognizers on bibliographical texts [2], Stanford NER was identified as the best system. Another evaluation on microposts [12] shows that Stanford NER is the second best after OpenCalais. This last result is especially relevant for our context, since we also have very short texts as input. Since OpenCalais is a commercial product, Stanford NER was thus our best choice. Note that this tool not only detects named entities mentions in a text, but also disambiguates them according to one of the following classes : PERSON, ORGANIZATION and LOCATION.

2. https ://opennlp.apache.org/
3. http ://alias-i.com/lingpipe/
4. http ://www.opencalais.com/

5.2.2 Semantic annotation

Most of the semantic annotators available as online services are commercial products. For our experiments, we decided to use non-commercial systems only.

DBpedia Spotlight [21] is a semantic annotator that uses a two-step process : first, entity mentions are spotted in text and then linked to some entity in DBpedia. The spotting phase relies on a set of surface forms extracted from DBpedia (titles, redirects) and anchors of Wikipedia links. To achieve entity linking, DBpedia Spotlight pre-ranks entity candidates for each surface form spotted in the text. It combines a prior score and a contextual score to determine which candidate entity is the most relevant. The prior score represents an estimation of how often the surface form is used as an anchor in a Wikipedia hyperlink that points to the entity page. Formally it corresponds to the probability $P(e|s)$, where e and s are the entity and the surface form, respectively. The contextual score takes into account the context of the phrase (a window of words around the phrase) and the context of each candidate entity. Note that DBpedia Spotlight has a confidence parameter that can be set whose value is between 0 and 1 (default value is 0.5). The effect of this parameter is to remove some annotations. The highest the confidence value, the highest is the probability of eliminating an annotation if it has more than one candidate for entity linking.

Similarly to Spotlight, Tagme [13] uses a list of surface forms extracted from Wikipedia anchors. For entity linking, it is based on a collective disambiguation process where, for each candidate entity of a surface form, a score of relatedness with the candidates of other surface forms is computed. The selected candidate is the one that maximizes a final score that combines all these relatedness scores. Tagme also uses a pruning method that retains only annotations that have a high link probability (defined as the number of Wikipedia articles that use it as an anchor, divided by the number of articles that mention it) or that have a high coherence score, which is computed by comparing it to all other annotations (by averaging over relatedness scores). Note that Tagme has been developed to be efficient with short texts, such as micro-blog posts.

Another semantic annotator that has been proposed recently is Babelfy [23], which uses a graph-based approach. Surface forms in text are associated to one or more vertices in BabelNet [24], a semantic network built from Wikipedia and Wordnet. Disambiguation is then achieved by a process that identifies the densest sub-graph according to some coherence metric.

Finally, AIDA [17] is a semantic annotator that assigns, to each spotted surface form, a value that corresponds to the prior score in DBpedia Spotlight, and a context similarity

score. These scores are weighted using links and information extracted from DBpedia and YAGO. AIDA uses an additional score that estimates the coherence between two entities. This score is calculated using the Wikipedia inlinks of each entity. The main contribution of this system is a graph-based algorithm. The graph is composed of text mentions nodes and entities nodes. AIDA extracts the sub-graph which has the highest density, where the density is a combination of the three computed scores for each annotation.

5.2.3 Entity typing

None of the previous systems achieves the full task of the OKE Challenge, which also includes entity typing according to some predefined types. Stanford NER does not consider the ROLE type. Semantic annotators only identify named entities in DBpedia, without typing them according to the OKE challenge type nomenclature. The best performing system for this task in the 2015 OKE Challenge [26] is ADEL [29], whose F-score is 0.60. It uses a hybrid approach that combines linguistic and semantic features. Linguistic resources, such as POS tagger, gazetteer and named entity recognizer, are used for the entity recognition task. For entity linking, some filtering is made, based on inbound and outbound links in Wikipedia. To select the most relevant entity among candidates, a graph-based approach similar to Babelfy is used. For typing with `dul:Role` class, ADEL mainly relies on a gazetteer (containing lists of occupations and nationalities). For other types, a manual alignment is built. Finally, a classifier is trained to filter out irrelevant entities. The second best-performing system, FOX [33], has a F-score of about 0.5. Typing is achieved by using a pattern-based process that identifies the segment that expresses the type of the entity in the input. One important advantage of FOX is that it is available through the GERBIL platform, thus facilitating peformance comparison.

5.3 System implementation

5.3.1 System Architecture

WESTLAB is composed of three modules, as illustrated in Figure 5.1 : 1) entity spotting, which consists of identifying relevant mentions in the text, 2) entity linking, where mentions are associated to some entity in DBpedia, when possible, and 3) entity typing, where the system tries to find, for each entity, the corresponding type in the DUL ontology. We will now describe each module.

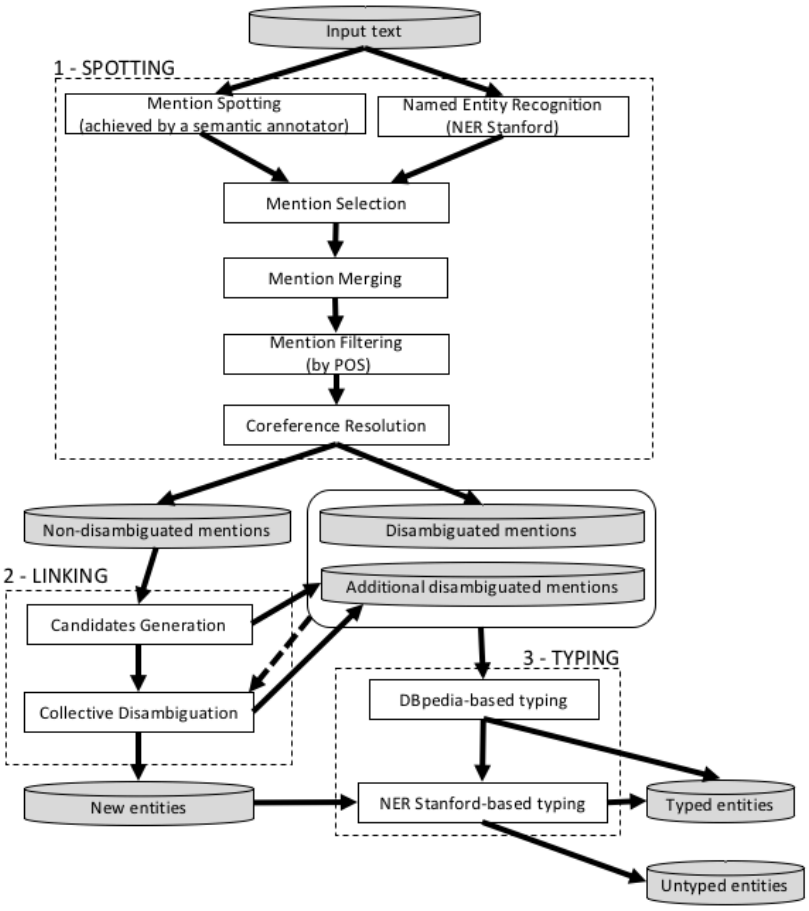


Figure 5.1 System architecture

5.3.2 Entity Spotting

The goal of this module is to extract all relevant mentions that can be found in the input text. We relied on semantic annotators freely available as web services and one named entity recognizer, the Stanford NER [14]. We experimented with four semantic annotators : DBpedia Spotlight, Babelfy, AIDA and Tagme. These semantic annotators not only detect relevant mentions in text, but also identify a corresponding entity in DBpedia. Stanford NER detects named entities that are tagged according to one of the following classes : *person*, *organization* and *location*. Each of these types can be aligned to a type in the OKE challenge : *Person*, *Organization* and *Place*. However, the type *role*, which is also an OKE challenge type, is not recognized by the Stanford NER.

After running these semantic annotators and Stanford NER, we noticed a frequent overlap between mentions detected by semantic annotators and named entities extracted by Stanford NER. In this case, we keep the longest mention. Consider the following example, where two

mentions found by DBpedia Spotlight are indicated in boldface :

> *John Stigall received a Bachelor of arts from the* ***State University*** *of* ***New York*** *at Cortland*

In this case Spotlight annotates *State University* and *New York* separately, whereas Stanford NER recognizes *State University of New York* as a single named entity. We select this last mention and discard the two separate mentions returned by DBpedia Spotlight. This is done at the *Mention selection* step.

At this stage, we still do not always have the correct mention. In fact, in this example, the correct mention is *State University of New York at Cortland*. To deal with this problem, we developed a *mention merging algorithm*. Given a mention, the algorithm attempts to expand it to cover the next mention. Such expansion is permitted only if the second mention immediately follows the first one or if the mentions are separated by one of these patterns : a word, a comma, a comma followed by a word, a period or a period followed by a word. Following this rule, in the previous example the mention *State University of New York* will be expanded into *State University of New York at Cortland*. This process is repeated until we reach a state where the mention cannot be further expanded. The expanded mention obtained at each step is memorized, such that at the end we obtain a list of mentions that are ordered from the longest one to the shortest one. We then select the first mention in this list for which there is an entity in DBpedia whose label corresponds to this mention.

The next step is *mention filtering*, where the POS of each word in a mention is identified by using the Stanford POS tagger. Every mention that contains a verb is removed from the list. Finally, we use Stanford Coreference resolution to find, for each pronoun, the coreferent mention. The mention and its co-referents are thus linked to the same entity.

The output of this module consists in two sets. One set contains the mentions detected by the semantic annotator. These mentions are already disambiguated with their corresponding entities in DBpedia. The other set contains the named entities returned by Stanford NER and the new mentions that resulted from the merging algorithm. Remember that every mention that overlaps another mention is removed if the other one is longer. The second set does not contain any disambiguated mention and must be processed by the following module, which performs entity linking for these mentions. Note that mentions already detected by the semantic annotator are not added to this set.

5.3.3 Entity Linking

For every non-disambiguated mention, we query DBpedia to extract the entity whose label corresponds to the mention. If there is no such entity, we create a new URI resource as described in the OKE challenge. If there is one result, we check if it corresponds to a disambiguation page. If so, all entities that are referred by this disambiguation page are taken as candidates, and are given as input to the next step, that is, the disambiguation process. If the entity corresponds to a normal page (i.e not a disambiguation page), the entity is linked to the mention, and immediately added to the set of disambiguated entities.

When we have more than one candidate, the selection of the entity that will be linked to the mention is achieved by taking into account the other mentions that have already been disambiguated in the text. In this case, non-disambiguated mentions are processed sequentially. For each mention we compute a score for each candidate c associated to this mention. This score is based on the "outlinks" of the corresponding Wikipedia article of the entity c and is defined as follows :

$$Score(c) = \frac{\sum_{e_i \in G} outlink(e_i) \cap outlink(c)}{outlink(c) \cup outlink(G)}$$

where G is the set of all distinct entities that are already disambiguated.

We keep the candidate with the highest score value, and thus obtain a new linked mention that is added to the set of disambiguated entities. The process iterates until every mention has been disambiguated.

5.3.4 Entity Typing

The goal of this step is to align the extracted entities with one of the following classes in the DUL ontology : DUL :Person, DUL :Organization, DUL :Place and DUL :Role. If a mention is not linked to a DBpedia entity, it is necessarily a mention that was recognized by Stanford NER. In this case, we simply apply the following mapping :

— Stanford :ORGANIZATION → DUL :Organization
— Stanford :LOCATION → DUL :Place
— Stanford :PERSON → DUL :Person

For mentions associated with a DBpedia URI, we try to find the type of the linked entity by executing the following SPARQL query :

```
SELECT ?type WHERE {<entity> rdf:type ?type }
```

If the query returns with success, the following mapping is applied, according to the instantiation of the variable `?type` :

— dbo :organisation → DUL :Organization
— yago :Organization108008335 → DUL :Organization
— dbo :Place → DUL :Place
— dbo :Location → DUL :Place
— dbo :EthnicGroup → DUL :Person
— dbo :Person → DUL :Person
— foaf :person → DUL :Person

If no type can be identified with this first query, we execute a second SPARQL query, where we check if the relevant information is represented by a predicate :

```
SELECT ?predicate WHERE {[] ?predicate <entity>}
```

According to the value extracted for the predicate, the following mapping is used :

— dbo :affiliation → DUL :Organization
— dbp :owner → DUL :Organization
— dbp :office → DUL :Organization
— dbo :birthPlace → DUL :Place
— dbo :location → DUL :Place
— dbo :deathPlace → DUL :Place
— dbo :occupation → DUL :Role
— dbp :occupation → DUL :Role

A conflict can appear with this method, when predicates are extracted with different types. This kind of conflict especially occurs between types *DUL :Role* and *DUL :Organization*, as for *State_University_of_New_York*, for which both predicates *dbo :affiliation* and *dbo :occupation* are extracted. In this case we count the number of occurrences of each predicate, and select the one with the highest number of occurrences. If no type can be identified with this second query, we execute a third SPARQL query, where we check if the relevant information is represented by a predicate, with the entity as subject :

```
SELECT ?predicate WHERE {<entity> ?predicate []}
```

According to the value extracted for the predicate, the following mapping is used :

— geo :geometry → DUL :Place

If this third query doesn't help to identify the type, we check if the mention was detected by Stanford NER in the spotting phase, and if so we use the mapping given at the beginning of this section. If not, the entity remains untyped.

5.4 Experiments and results

5.4.1 Training set description

We conducted our experiments against the training set distributed by the OKE challenge organizers, which is composed of 195 manually annotated sentences. It contains 1030 mentions annotated with 683 distinct entities, among which 530 are linked to DBpedia and 153 are defined as NIL mentions (URIs that do not exist on DBpedia and are relevant). These new entities are used to populate knowledge bases with new resources. Table 5.1 provides detailed statistics on this dataset, for each type of DUL entity.

5.4.2 Results on the OKE challenge training dataset

To evaluate the potential of our approach, we applied it on the training set, using four publicly available semantic annotators : DBpedia Spotlight, Babelfy, Tagme and AIDA. We used the appropriate configuration of each system in order to obtain the highest number of possible mentions. For DBpedia Spotlight, we set the confidence parameter at 0.0, while for Babelfy we use the default configuration, which does not limit the spotting phase to named entities only. We also rely on the default configuration for Tagme. For the purpose of our evaluation, we developed our own evaluation script following the OKE challenge criteria mentioned on the homepage of the competition. Only exact matches are counted. The macro-averaged precision-recall results of the WESTLAB system coupled with each semantic annotator are shown in Table 5.2. As we can see, the best performance was obtained by our framework coupled with Babelfy, in the three subtasks. However the difference with the F-scores obtained using the other semantic annotators is very low.

Table 5.1 Statistics on the training set per DUL type

Type	# mentions	# disamb. mentions	# NIL mentions
Dul :Role	165	144	21
Dul :Organization	237	198	39
Dul :Person	446	342	104
Dul :Place	182	171	11
Total	1030	855	175

Table 5.2 Results of the entity recognition, linking and typing tasks on the training dataset

	Entity Recognition			Entity Linking			Entity Typing			
	P	R	F	P	R	F	P	R	F	Average F
DBpedia Spot.	0.7388	**0.8430**	0.7875	0.5923	0.7224	0.6510	0.6538	**0.7498**	0.6985	0.7123
Babelfy	0.7742	0.8050	**0.7894**	0.6460	0.7091	0.6760	0.7031	0.7393	**0.7207**	**0.7287**
Tagme	0.7146	0.8153	0.7616	0.5912	**0.7317**	0.6540	0.6377	0.7315	0.6814	0.6990
AIDA	**0.8652**	0.6890	0.7671	**0.7670**	0.6273	**0.6901**	**0.7981**	0.6404	0.7106	0.7226

5.4.3 Baseline comparisons

We compared WESTLAB with Stanford NER and each semantic annotator used in our evaluation as a baseline. To avoid improper comparison, we conducted two different evaluations. The first one compares WESTLAB with Stanford NER to evaluate its capabilities for the subtasks of entity recognition and typing. In the second evaluation, we compare WESTLAB with each of the employed semantic annotators to assess their initial performance in extracting and disambiguating mentions. Since the best performances are obtained when WESTLAB is coupled with Babelfy, as shown in Table 2, this annotator has been chosen in our following experiments.

Comparison with Stanford NER

Since Stanford NER uses 3 classes only (Organization, Person and Place) we removed Role mentions from the training dataset for this particular evaluation. The results are listed in Table 5.3. We can note that WESTLAB highly outperforms Stanford NER.

Comparison with semantic annotators

We compared WESTLAB with the four semantic annotators used individually, under various configurations. For DBpedia Spotlight, we experimented with two confidence scores : 0.5 and 0.0. For Babelfy, we tested a configuration where only named entities are extracted. Finally,

Table 5.3 Comparison with Stanford NER

	Entity Recognition			Entity Typing			
	P	R	F	P	R	F	Average F
WESTLAB	0.8395	0.8183	0.8288	0.7923	0.7754	0.7837	**0.8063**
NER Stanford	0.8120	0.6740	0.7360	0.7360	0.6130	0.6690	0.6745

for Tagme, we also experimented a configuration that extracts mentions not linked to any entity (Tagme all extractions). The results are given in Table 5.4 and clearly show that our approach outperforms all the others.

Compared to semantic annotators, the increase in performance is at least 20 % for the F-score. Note that the improvement is more evident for the subtask of entity recognition.

5.4.4 Result on the OKE Challenge evaluation dataset

For the OKE challenge 2016, the systems were evaluated against a new dataset, composed of 55 hand-annotated sentences. Table 5.5 provides details about the dataset per Dul type. Note that the ratio of disambiguated mentions does not significantly differ from the training dataset. Regarding the distribution of types, the situation is different : type *Dul :Person* is more dominant in the training dataset (43% instead of 31%) and there is no instance of NIL mentions for type *Dul :Place*.

The results of WESTLAB, coupled with each individual semantic annotator, are given in Table 5.6. These results confirm that the best performance for our system is obtained using Babelfy.

To compare WESTLAB with the semantic annotators taken separately, we used GERBIL [37]. We tested the performances for the entity recognition and linking subtasks only, since the existing systems do not provide a DUL type. The results, presented in Table 5.7, show that WESTLAB, coupled with Babelfy, still outperforms these baseline systems.

There were two competing systems in the OKE challenge 2016 : WESTLAB and a new version of Adel [30], the winner of the 2015 OKE challenge. The results are shown in Table 5.8. Note that we report the results against the original dataset used during the challenge (original gold standard), and against a corrected gold standard modified by the competition

Table 5.4 Comparison with semantic annotators

	Entity Recognition			Entity Linking			
	P	R	F	P	R	F	Average F
WESTLAB	0.7742	0.8050	0.7894	0.6460	0.7091	0.6760	**0.7327**
Dbpedia Spotlight 0.0	0.3620	0.6370	0.4620	0.2650	0.5470	0.3570	0.4095
Dbpedia Spotlight 0.5	0.6570	0.4450	0.5310	0.5760	0.4630	0.5130	0.5220
AIDA	0.8245	0.5530	0.6620	0.7230	0.5160	0.6020	0.6320
Babelfy	0.3100	0.7340	0.4360	0.3250	0.6360	0.4330	0.4345
Babelfy NED	0.6500	0.4990	0.5640	0.6070	0.5470	0.5750	0.5695
Tagme	0.5110	0.7540	0.6090	0.4120	0.7060	0.5200	0.5645
Tagme (all extractions)	0.5080	0.7570	0.6080	0.4120	0.7060	0.5200	0.5640

Table 5.5 Statistics on the evaluation set per DUL type

Type	# mentions	# disamb. mentions	# NIL mentions
Dul :Role	86	71	15
Dul :Organization	105	91	14
Dul :Person	105	82	23
Dul :Place	44	44	0
Total	340	288	52

Table 5.6 Results of the entity recognition, linking and typing tasks on the evaluation dataset

	Entity Recognition			Entity Linking			Entity Typing			
	P	R	F	P	R	F	P	R	F	Average F
DBpedia Spot.	0.7737	0.6235	0.6906	**0.8255**	0.5147	0.5784	0.5853	0.5853	0.5853	0.6366
Babelfy	**0.744**	**0.7353**	**0.7396**	0.764	**0.5618**	**0.6475**	**0.6441**	**0.6441**	**0.6441**	**0.6771**
Tagme	0.6606	0.75	0.7025	0.749	0.5618	0.642	0.6412	0.6412	0.6412	0.6619
AIDA	0.8204	0.5912	0.6872	0.801	0.4735	0.5952	0.55	0.55	0.55	0.6108

Table 5.7 Comparison with semantic annotators

	Entity Recognition			Entity Linking			
	P	R	F	P	R	F	Average F
WESTLAB	0.7448	0.7382	0.7415	0.761	0.5618	0.6464	**0.6939**
Dbpedia Spotlight	0.6042	0.3412	0.4361	0.931	0.3176	0.4737	0.4549
AIDA	0.8706	0.4353	0.6804	0.5803	0.3676	0.5133	0.5968
Babelfy	0.463	0.4235	0.4424	0.7972	0.3353	0.472	0.4572
Tagme	0.6192	0.55	0.5826	0.8492	0.4971	0.6271	0.6048

organizers to fix some annotations issues after the challenge (corrected gold standard). On this corrected gold standard, the results of Adel are missing since this system does not offer a public web service to test it. The complete results can also be found at the challenge homepage [5]. Note that the evaluation uses a weak match, that is, an annotation is true if and only if it overlaps the annotation of the gold standard. Thus a weak annotation match does not require an exact match. As we can notice, WESTLAB outperforms Adel in all micro and macro measures.

5.4.5 Analysis of errors

We analyze the errors generated by our system and discuss them in this section.

Spotting errors

Table 5.9 shows some statistics on the evaluation dataset. 71% of the mentions are returned by Babelfy (240 out of 337). 27 mentions among the incorrect ones should not be annotated and are due to the limits of our approach. We judge that 26 mentions are missing in the gold standard and should not be considered as errors made by WESTLAB (for example, WESTLAB spots *Scotland* in sentence *In October 1850 ... Maxwell left Scotland for the University of Cambridge*, and this mention is not in the gold standard). 8 errors are incomplete mentions (e.g. *Minister* instead of *Minister of Education*) or mentions that should be splitted (e.g. *bishop of Bologna* should be splitted into two mentions, *bishop* and *Bologna*). In fact, as we explained in section 5.3.2, when there are overlapping mentions, we always keep the longest one. For example, we may obtain the mention *Paris, France*, whereas the gold standard contains two separate mentions, *Paris* and *France*. To avoid this kind of errors, some heuristics

5. https ://github.com/anuzzolese/oke-challenge-2016

Table 5.8 Results on the evaluation dataset at the OKE challenge 2016, with weak match

Original gold standard

	Micro F1	Micro P	Micro R	Macro F1	Macro P	Macro R
WESTLAB (using Babelfy)	**0.6676**	**0.6964**	**0.6509**	**0.6516**	**0.6902**	**0.6439**
Adel	0.6249	0.6689	0.5942	0.6064	0.6606	0.5846

Corrected gold standard

	Micro F1	Micro P	Micro R	Macro F1	Macro P	Macro R
WESTLAB (using Babelfy)	**0.6998**	**0.7402**	**0.6696**	**0.6926**	**0.7449**	**0.6753**

must be added to determine when a mention must be splitted. However, after a careful analysis of the training dataset, we did not find any clear rule governing the decision of splitting a mention or not.

For incorrect mentions originating from Stanford NER, which represent 16% of the total, 9 are named entities involving a modifier, as in ***Zionist*** *affairs* and ***American*** *high schools*, whereas the gold standard does not contain modifiers, i.e. *high schools* is annotated individually. As a solution, these mentions could be filtered out using some heuristics based on syntactic parsing. Additionnally, 4 mentions returned by the Stanford NER should be splitted according to the gold standard (e.g. *Principal of Marischal* should be splitted into *Principal* and *Marischal* separately).

13% of the mentions spotted by Stanford NER are coreference mentions (personal pronouns). 7 of these mentions are not annotated in the evaluation dataset, due to wrong offsets.

Regarding the missing mentions (89 out of 340), which decrease WESTLAB's recall performance, about half of them are due to the weakness of WESTLAB in detecting roles. Roles are usually expressed by common names that are not easily detected by semantic annotators. We also note that the problem about splitted mentions discussed earlier also affects recall : 23% of the missing mentions are caused by this problem. Taking the example given previously, the two mentions *Principal* and *Marischal* would be considered as missing, since WESTLAB only detects *Principal of Marischal*.

Disambiguation errors

Table 5.10 shows the disambiguation errors made by our system. The results are calculated only for correctly spotted mentions. "Stanford NIL" represents the mentions annotated as NIL mentions. Considering the erroneous mentions generated by the annotator, we found that most of them (74%) are real errors made by Babelfy. The remaining ones are correctly disambiguated mentions but either annotated as NIL mentions in the gold standard (for example, *author* is annotated by WESTLAB with *http ://dbpedia.org/resource/Author* while it is annotated as NIL in the gold standard) or not found in the gold standard. Now

Table 5.9 Spotting errors, according to their origin.

Annotator (240)		Stanford (54)		Coreference (43)		Total (337)
Correct	Wrong	Correct	Wrong	Correct	Wrong	
179	61	36	18	36	7	

Table 5.10 Disambiguation errors

Annotator (190)		Collective Disambiguation (29)		Stanford NIL (32)		Total (251)
Correct	Wrong	Correct	Wrong	Correct	Wrong	
145	45	23	6	22	10	

considering the collective disambiguation process, most of the mistakes (4 out of 6) are made because the right entity is missing in the list of candidates. Note that the candidates generator is based on an exact match of the label associated to the DBpedia entity. A partial match could be more appropriate but we must determine the threshold of similarity, and this will have a cost in terms of performance. Another source of error is the fact that our algorithm is based on the outlinks of each entity. Sometimes there are not enough mentions already disambiguated by the semantic annotator, and the number of outlinks is not sufficient to identify the right candidate.

For the incorrect Stanford NIL mentions, we found two sources of errors. In some cases, our candidate generator does not return any candidate for the target mention. The other cause is the limits of the Stanford coreference resolution which sometimes does not identify correctly a co-referent.

Typing errors

Table 5.11 shows some statistics about the errors made when typing the annotated mentions according to DUL categories. Only 12 % among the mentions that are correctly spotted are incorrectly typed. For mentions that come from the annotator, 14 mistakes are due to the priority rule used in our implementation (i.e. Person>Place>Organization>Role). In fact, DBpedia entities might be typed with more than one of the four types used in this task. For example, *National_Academy_of_Sciences* is typed as a *dbo :place* and a *yago :Organization108008335*. To select the type, our current system simply gives priority to *dbo :Place*. In the other 14 cases, we got the wrong type simply because the entity selected during the disambiguation process is the wrong one. For the mentions that come from Stanford, the 3 errors are due to the Stanford NER, since the type it provides is directly returned by WESTLAB.

Table 5.11 Typing errors

DBpedia (202)		Stanford (49)	
Correct	Wrong	Correct	Wrong
174	28	46	3

5.5 Conclusion

In this paper, we have shown that combining the outputs of a named entity recognizer and a semantic annotator, and applying a collective disambiguation approach for selecting an entity among candidates, outperforms these two systems taken separately. We have also shown that by querying DBpedia to extract explicit and implicit types of entities, and using some manually defined mapping to the restricted set of four types used in the challenge, we also outperform the systems that participated to the 2016 OKE Challenge. Our approach is generic and does not depend on the dataset. In terms of efficiency, the most time-consuming step is querying DBpedia, which is used for typing.

For future work, we plan to improve the linking process and define a new approach to combine multiple annotators. We also plan to develop a new typing approach which uses the hierarchy of the DBpedia ontology instead of our current manual alignement with the DUL ontology.

5.6 Aknowledgement

This research has been funded by the NSERC Discovery Grant Program.

CHAPITRE 6 DISCUSSION GÉNÉRALE

Dans les deux chapitres précédents, nous avons proposé deux approches pour améliorer des annotateurs sémantiques. La différence entre les deux systèmes, FICLONE et WESTLAB, réside dans le fait que nous avons testé l'amélioration d'un seul système pour FICLONE ("DBpedia Spotlight") et de quatre pour WESTLAB, selon une méthode générique (DBpedia Spotlight, AIDA, Tagme et Babelfy). D'après nos résultats, FICLONE convient davantage à des textes longs qu'à des textes courts (voir tableau 4.7). Dans le cadre de la compétition Open Knowledge Extraction, les données de test utilisées se composent de phrases courtes. Pour l'OKE, nous avons choisi d'utiliser Babelfy car, comme le montre le tableau 5.2, c'est avec ce dernier que nous avons obtenu les meilleurs résultats, lors de nos tests préliminaires sur les données d'entraînement.

Dans ce mémoire, nous nous sommes intéressés à plusieurs aspects de la tâche d'annotation sémantique. Tout d'abord, nous avons investigué à quel niveau les annotateurs existants se trompaient. Ensuite, nous avons tenté d'améliorer les performances des annotateurs sémantiques au travers de l'ajout d'un mécanisme de détection d'entités nommées et l'utilisation d'un mécanisme de désambiguisation collective. Par le biais des deux articles présentés, nous avons démontré que les annotateurs génèrent des erreurs au niveau de la détection des mentions et de leur désambiguïsation. Pour y remédier, nous avons mis en évidence l'importance d'introduire un outil de reconnaissance d'entités nommées pour faire ressortir plus de mentions et en corriger quelques-unes qui étaient initialement incorrectes, ainsi que l'utilisation de différents scores pour l'étape de désambiguïsation, le commonScore pour le système WESTLAB, le directScore et le score Jaccard pour FICLONE. En ce qui concerne la phase de génération des candidats, DBpedia Spotlight était le seul annotateur que nous étions capables de tester en faisant appel à son service Web "Candidates". Les erreurs que nous avons constatées à cette étape ont été contournées par l'utilisation d'une source externe (LRD&WAT), que nous avons présentée dans le chapitre 4.

Au-delà des erreurs que renvoient les annotateurs, avec WESTLAB nous avons pu augmenter leurs performances. En effet, en expérimentant sur l'ensemble de données de tests fournies par les organisateurs de la compétition, nous avons enregistré une amélioration sur le score F1 : Tagme de 0.56 à 0.71, DBpedia Spotlight de 0.41 à 0,72, AIDA de 0.63 à 0.73 et Babelfy de 0,57 à 0.73. Chacun des progrès que nous venons de citer provient du tableau 5.2, en prenant la moyenne des scores F1 pour l'"Entity Recognition" et "Entity Linking".

Au sujet de FICLONE, nous avons réussi à accroitre les performances actuelles de DBpedia

Spotlight pour la tâche d'annotation sémantique (avec FICLONE SA) d'une manière générale (tableau 4.8), ainsi que pour le cas particulier de la désambiguïsation des entités nommées (avec FICLONE NED) (tableau 4.7). Pour les deux systèmes (FICLONE et WESTLAB), les annotations collectées directement des annotateurs sont automatiquement utilisées comme une entrée dans notre algorithme. Pourtant, comme le démontrent les tableaux 5.10 et 4.12, ces annotations ne sont pas toujours correctes. Cela a une influence sur les annotations que va générer notre approche.

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS

Ce mémoire présente deux approches qui servent à améliorer les performances des annotateurs sémantiques. Ces approches optimisent les trois phases de l'annotation sémantique : la détection des mentions, la génération des candidats et la désambiguïsation. L'approche développée dans WESTLAB est générique et permettant une augmentation significative. Indépendamment du système utilisé, cette méthode obtient un F1-Score supérieur à 70%. En ce qui concerne la méthode présentée dans FICLONE est spécifique à DBpedia Spotlight et ne peut pas être appliqué à d'autre annotateur sémantique, puisque DBpedia Spotlight est le seul système parmi ceux que nous avons utilisés tout au long de nos recherches qui offre un paramètre qui facilite la détection des mentions qui ont été mal désambiguïsées (*confidence*).

Les différentes expérimentations que nous avons conduites nous ont permis d'affirmer que nos approches sont prometteuses et que l'amélioration des annotateurs est toujours possible. Toutefois, il existe encore des pistes que nous pouvons explorer. Dans le cas de WESTLAB, nous avons pu filtrer les mentions non pertinentes grâce à une méthode de filtrage par type, alors que pour FICLONE nous n'avons pas eu cette possibilité. La première piste à explorer serait le perfectionnement du détecteur de mentions, en approfondissant les mentions générées par Stanford NER et l'annotateur sémantique. Concernant le module de génération des candidats, il faudrait, en premier lieu, assigner pour chaque candidat e_i d'une mention m donnée un score qui pourrait être utile pour la phase de désambiguïsation, comme par exemple, la probabilité à priori qui est implémentée dans la plupart des systèmes actuels. Il faudrait également ajouter une méthode dynamique pour générer des candidats pour les mentions qui, pour l'instant, n'existent pas dans l'ensemble de formes de surface que nous utilisons. Comme par exemple *T. Moody, C. Walsh*, ou la mention *Reggi Blinker* qui fait référence à "Regi_Blinker".

Enfin, pour la désambiguïsation, le plus important serait de vérifier les annotations retournées par l'annotateur sémantique. L'algorithme que nous avons implémenté en dépend fortement. En conséquence, afin de réduire cette forte dépendance, il faudrait calculer le score contextuel et le combiner avec les scores que nous avons mis en place. La solution idéale serait d'implémenter des méthodes d'apprentissage automatique pour calculer la similarité entre le contexte dans lequel apparaît une mention donnée et le contexte d'un candidat qui lui est associé.

RÉFÉRENCES

[1] E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Association for Computational Linguistics, 2009.

[2] S. Atdağ and V. Labatut. A comparison of named entity recognition tools applied to biographical texts. In *Systems and Computer Science (ICSCS), 2013 2nd International Conference on*, pages 228–233. IEEE, 2013.

[3] V. Bryl, C. Bizer, and H. Paulheim. Gathering alternative surface forms for DBpedia entities.

[4] R. C. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, volume 6, pages 9–16, 2006.

[5] M. Chabchoub, M. Gagnon, and A. Zouaq. FICLONE : improving DBpedia spotlight using named entity recognition and collective disambiguation, Natural Language Engineering. (soumis).

[6] M. Chabchoub, M. Gagnon, and A. Zouaq. *Semantic Web Challenges : Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, chapter Collective Disambiguation and Semantic Annotation for Entity Linking and Typing, pages 33–47. Springer International Publishing, Cham, 2016.

[7] E. Charton, M.-J. Meurs, L. Jean-Louis, and M. Gagnon. Semlinker system for kbp2013 : A disambiguation algorithm based on mutual relations of semantic annotations inside a document. In *Text Analysis Conference KBP. US National Institute of Standards and Technology (NIST)*, 2013.

[8] R. L. Cilibrasi and P. M. Vitanyi. The google similarity distance. *IEEE Transactions on knowledge and data engineering*, 19(3) :370–383, 2007.

[9] M. Cornolti, P. Ferragina, and M. Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 249–260. ACM, 2013.

[10] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 7, pages 708–716, 2007.

[11] L. Derczynski, D. Maynard, G. Rizzo, M. van Erp, G. Gorrell, R. Troncy, J. Petrak, and K. Bontcheva. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2) :32–49, 2015.

[12] S. Dlugolinsky, M. Ciglan, and M. Laclavik. Evaluation of named entity recognition tools on microposts. In *Intelligent Engineering Systems (INES), 2013 IEEE 17th International Conference on*, pages 197–202. IEEE, 2013.

[13] P. Ferragina and U. Scaiella. Tagme : on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[14] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[15] S. Hakimov, S. A. Oto, and E. Dogdu. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proceedings of the 4th international workshop on semantic web information management*, page 4. ACM, 2012.

[16] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. Kore : keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 545–554. ACM, 2012.

[17] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[18] P. Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901.

[19] T. Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[20] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4) :885–916, 2013.

[21] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia spotlight : shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.

[22] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[23] A. Moro, A. Raganato, and R. Navigli. Entity linking meets word sense disambiguation : a unified approach. *Transactions of the Association for Computational Linguistics*, 2 :231–244, 2014.

[24] R. Navigli and S. P. Ponzetto. Babelnet : The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193 :217–250, 2012.

[25] E. P. A. N'Techobo, A. Zouaq, and M. Gagnon. Semantic annotation for the analysis of political debates : A graph-based approach. In *International Conference on the Advances in Computational Analysis of Political Text*, Dubrovnik, 2016.

[26] A. G. Nuzzolese, A. L. Gentile, V. Presutti, A. Gangemi, D. Garigliotti, and R. Navigli. *Semantic Web Evaluation Challenges : Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, chapter Open Knowledge Extraction Challenge, pages 3–15. Springer International Publishing, Cham, 2015.

[27] A. Olieman, H. Azarbonyad, M. Dehghani, J. Kamps, and M. Marx. Entity linking by focusing DBpedia candidate entities. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 13–24. ACM, 2014.

[28] F. Piccinno and P. Ferragina. From tagme to wat : a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 55–62. ACM, 2014.

[29] J. Plu, G. Rizzo, and R. Troncy. *Semantic Web Evaluation Challenges : Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, chapter A Hybrid Approach for Entity Recognition and Linking, pages 28–39. Springer International Publishing, Cham, 2015.

[30] J. Plu, G. Rizzo, and R. Troncy. Enhancing entity linking by combining NER models. In *Open Knowledge Extraction challenge at ESWC 2016*, 2016.

[31] A. Pohl. Improving wikipedia miner word sense disambiguation algorithm. In *FedCSIS*, pages 241–248, 2012.

[32] M. Röder, R. Usbeck, S. Hellmann, D. Gerber, and A. Both. N3-a collection of datasets for named entity recognition and disambiguation in the NLP interchange format. *9th LREC*, 2014.

[33] M. Röder, R. Usbeck, R. Speck, and A.-C. N. Ngomo. *Semantic Web Evaluation Challenges : Second SemWebEval Challenge at ESWC 2015, Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, chapter CETUS – A Baseline Approach to Type Extraction, pages 16–27. Springer International Publishing, Cham, 2015.

[34] R. S. Sinha and R. Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *ICSC*, volume 7, pages 363–369, 2007.

[35] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago : A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[36] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[37] R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, et al. GERBIL : General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1133–1143. ACM, 2015.

[38] I. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence : an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30, 2008.

[39] L. Zhang and A. Rettinger. X-lisa : cross-lingual semantic annotation. *Proceedings of the VLDB Endowment*, 7(13) :1693–1696, 2014.