Introduction au Traitement Automatique de la Langue

Coréférences d'entités nommées

Carl Goubeau

15 décembre 2013

1 Introduction

La reconnaissance d'entités nommées est une tâche importante d'extraction d'information dans des corpus. Ces objets textuels correspondent à des catégories telles que des noms de personnes, d'organisations, de lieux, des dates, des montants... La coréférence est le phenomène qui consiste à designer le même objet par différentes expressions.

Après avoir collectivement annoté un corpus de texte avec les coréférences de chaque entité nommée, nous nous sommes intéressés au développement d'un programme en *Python* permettant une annotation automatique des coréférences.

Nous nous intéresserons, dans un premier temps, à un algorithme minimaliste. Nous Finirons par étudier le fonctionnement de la méthode d'extraction des coréférences améliorée.

2 Annotation automatique des coréférences

2.1 Méthode utilisée

Afin d'annoter les coréférences, nous définissons tout d'abord deux listes :
- entities : pour les entités trouvées
- values : pour les numéros des entités correspondantes

Par exemple, si l'entité "<LOCATION>USA</LOCATION>" est retrouvée en position 1 et 2 d'un texte, l'état des listes sera le suivant :

```
["<LOCATION>USA</LOCATION>"] # pour entities
[[1,2]] # pour values
```

Pour chacun des fichiers du corpus, nous recherchons à l'aide d'une expression régulière les différentes entités nommées.

```
regexp = re.compile("<[A-Z]+>[\w\.\,\s\-']+</[A-Z]+>")
```

Pour chacun des résultats que nous numérotons à l'aide d'une variable incrémentée, nous vérifions si l'entité a déjà été trouvée. S'il y a correspondance, nous ajoutons le numéros de l'entitée à la liste des coréférences déja trouvées, sinon, nous définissons une nouvelle entité.

L'entité est mémorisée avec son *"type"*. Cela nous permet de différencier par exemple *Paris* comme étant un prénom ou le nom de la ville.

2.2 Résultats obtenus

En utilisant le programme python qui compare les coréférences définies à la main et celles trouvées automatiquement par notre algorithme, nous trouvons les résultats suivants :

```
python evaluation-tp3.py clustersNE.ref etudiant4.ref
(55.01667422808361, 67.18844109695392, 60.08865164127736)
```

Qui correspondent respectivement aux valeurs moyennes pour la précision, le rappel et la f-mesure.

3 Amélioration des performances

3.1 Methodes utilisées

3.1.1 Entité nommée incluse dans une autre

Afin d'améliorer la performance de notre programme, un test sur l'inclusion des entités nommées a été ajouté.

Premièrement, cette partie vérifiait si l'inclusion, pour la valeur de l'entité nommée [1], était respectée dans un sens ou dans l'autre. Les résultats obtenus avec cette méthode sont les suivant :

```
python evaluation-tp3.py clustersNE.ref etudiant4.ref
(79.30501266071734, 81.3179115519278, 80.0532328009108)
```

Il sembla alors judicieux de vérifier également la correspondance du type de l'entité [2]. Cela augmenta légèrement la performance de l'algorithme :

```
python evaluation-tp3.py clustersNE.ref etudiant4.ref
(80.24159844629641, 83.55638709577248, 81.62294658933759)
```

3.1.2 Gestion des sigles

L'efficacité du programme peut être encore améliorée en prenant en compte les sigles présents dans le corpus. On notera que les acronymes sont définis avec des majuscules, c'est pourquoi une fonction permettant l'extraction d'un sigle depuis une chaîne de caractères sera utilisée par la suite.

Cette fonction procède de la manière suivante :

- elle utilise une expression régulière ne récupérant que les majuscules
- elle concatène les lettres obtenues dans une variable qu'elle retourne

1. par exemple pour "`<EN>entité</EN>`" on récupère la chaîne "entité"
2. par exemple pour "`<EN>entité</EN>`" on récupère la chaîne "EN"

Ainsi, pour les chaînes "USA", "United States of America" et "U.S.A." on obtiendra le sigle "USA".

Dans un premier temps, seul l'égalité des acronymes fut ajoutée à l'algorithme. Cela augmenta une nouvelle fois la qualité des coréférences obtenues :

```
python evaluation-tp3.py clustersNE.ref etudiant4.ref
(83.71884091417819, 84.14169360213158, 83.72836853479737)
```

Cependant, cette vision minimaliste de la gestion des sigles peut poser certains problèmes, par exemple :

```
>>> testSigle("<LOCATION>Ancenis</LOCATION>", "<LOCATION>Angers
  </LOCATION>")
>>> True
# Ancenis = Angers d'apres cette fonction
```

Différents tests ont été réalisés concernant la longueur minimale que doit avoir un sigle pour être considéré comme significatif. Les meilleurs résultats ont été obtenus à partir d'une longueur de 2 lettres dans l'acronyme.

Après réflexion, il sembla intéressant de vérifier l'inclusion d'un sigle dans un autre. Là encore, la longueur des chaînes traitées eu de l'importance. Ces tests ce sont révélés plus performants avec des sigles d'au moins 3 lettres.

3.2 Résultats finaux obtenus

Avec toutes ces améliorations apportées à l'algorithme initial, les résultats obtenus sont les suivants :

```
python evaluation-tp3.py clustersNE.ref etudiant4.ref
(85.97162541528486, 86.3748016573873, 86.00409949216233)
```

4 Conclusion

Les différentes améliorations apportées à la fonction de base ont permis d'obtenir une performance plutôt bonne pour l'annotation automatique des coréférences dans notre corpus. De plus, le fichier servant de référence peut contenir une certains nombre d'erreurs, de part l'ambigüité de certaines entités, et de part la contruction de ce document à la main.