Rapport de TP Génie Logiciel

Scrum Master
Diennet Thomas

Développeurs
Chaibainou Caroline
Makrini Zakari
Olivero Antoine
Parilla Valentin

6 janvier 2020

1 Abstract

Les travaux de recherche du Laboratoire Informatique d'Avignon (LIA) nécessitent la consultation de nombreux articles scientifiques dans le cadre de leur recherches.
Cependant, la lecture et le tri d'information est une tâche fastidieuse et grandement chronophage si elle est assignée à une personne physique.
L'enjeu de notre mission était de parvenir à automatiser ce travail de lecture et de tri d'information à l'aide de programmes informatiques, en respectant les contraintes de format de sortie (.txt / .xml).
La solution proposée par notre équipe afin de répondre à la demande du LIA est un programme en langage python, qui est capable de recevoir un corpus darticle scientifiques, en permettant à l'utilisateur de choisir spécifiquement ceux qu'il désire traiter s'il ne souhaite pas traiter l'intégralité du corpus. Le programme va alors créer un nouveau dossier, traiter les textes choisis pour en extraire les informations souhaitées par lutilisateur et créer de nouveaux fichiers pour chaque fichier traité, contenant le résultat du traitement, avec deux choix possible de type de fichiers de sortie : xml ou txt.

2 Méthode (explication du système)

2.1 Opérations réalisées par le système :

- Proposer via un menu la selection de PDF à traiter, ou l'intégralité du corpus
- Créer un nouveau dossier
  - Pour chaque fichier pdf contenu dans le dossier ciblé :
    - Convertir le fichier pdf à l'aide pdf2text
    - Si l'option choisie est .txt
      - - Créer un nouveau fichier .txt relatif au pdf
      - Récupérer le nom du fichier d'origine et l'écrire dans le fichier dans une ligne
      - Récupérer le titre du fichier d'origine et l'écrire dans le fichier dans une ligne
      - Récupérer le nom et l'adresse du/des auteur(s) et l'écrire dans le fichier dans une ligne
      - Récupérer le résumé ou abstract de lauteur du fichier dorigine et l'écrire dans le fichier
      - Récupérer les références bibliographiques de l'article et l'écrire dans le fichier
  - Si l'option choisie est .xml

— Créer un nouveau fichier .xml relatif au pdf
— A l'intérieur d'une balise <article> :
  — Récupérer le nom du fichier dorigine et l'écrire dans le fichier dans une balise <preamble>
  — Récupérer le titre du fichier dorigine et l'écrire dans le fichier dans une balise <titre>
  — Récupérer le nom et l'adresse du/des auteur(s) et l'écrire dans le fichier dans une balise <auteur>
  — Récupérer le résumé ou abstract de lauteur du fichier dorigine et l'écrire dans une balise <abstract>
  — Récupérer les références bibliographiques de l'article et l'écrire dans une balise <biblio>

2.2 Execution du système :

La fonction parserTXTsolo() nous montre l'exemple de comment nous opérons au traitement des textes en utilisant le système conçu.
Appelée après le choix des pdf à parser dans le menu, ainsi que le choix du format de sortie, cette fonction fait appel à toute les méthodes ayant le rôle de récupérer dans les fichiers convertis les sections de texte désirées.
Il est donc question pour chaque pdf de le lire, le traiter avec les différentes parser pour ensuite écrire éléments retenus dans un nouveau fichier se situant dans un dossier lui même crée à l'occasion par le système.

```
def parserTXTsolo(f1):
     f1 = f1.strip('.pdf')
     file_to_open = 'Papers/' + f1 + '.pdf'
     file_to_open = file_to_open.replace(' ', '\ ')
     file_to_read = 'ConvertedPapers/' + f1 + '.txt'
     temp = file_to_read
     file_to_read = file_to_read.replace(' ', '\ ')
     command = 'pdf2txt ' + file_to_open + ' > ' + file_to_read
     os.system(command)
     open_read = open(temp, 'r+')
     #Name
     parsed_file = 'ParsedPapers/' + f1 + '.txt'
     open_write = open(parsed_file, 'w+')
     open_write.write("Nom du fichier : \n \t"+f1+"\n")
     print(f1)
     #Title
     titre = getTitle(open_read)
     open_write.write("Titre de l'article : \n "+titre.replace('\n', ' ')+"...")
     #Auteur
     authors = getAuthor(open_read)
     open_write.write(" \nAuteurs : \n"+authors)
     #Abstract
     abstract = getAbstract(open_read)
     open_write.write("\nAbstract/Resume de l'article : \n"+abstract.replace('\n', ' ')[0 : 150]+"
     #Introduction
     intro = getIntro(open_read)
     open_write.write("\nIntroduction : \n\t"+intro.replace('\n', ' ')[0 : 150]+"...\n")
```

```
#Corps
corps = getCorps(open_read)
open_write.write("\nCorps : \n\t"+corps.replace('\n', ' ')[0 : 300]+"...\n")
#Conclusion
conclusion = getConclusion(open_read)
open_write.write("\nConclusion : \n\t"+conclusion.replace('\n', ' ')[0 : 150]+"...\n")
#Discussion
discussion = getDiscussion(open_read)
open_write.write("\nDiscussion : \n\t"+discussion.replace('\n', ' ')[0 : 150]+"...\n")
#Bibliography
bibliography = getBibliography(open_read)
open_write.write("\nBibliographie : \n\t"+bibliography.replace('\n', ' ')+"\n")

open_read.close()
open_write.close()
```

3 Résultats

Afin de tester la fiabilité de notre programme, nous l'avons soumis à un test de précision. Nous lui avons fait traiter un corpus de 9 fichiers PDF (nous avons testé la sortie .txt seule car le résultat est le même pour la sortie .xml), et avons observé la qualité de récupération des sections demandées. Le tableau ci-dessous contient les résultats relatifs à ce test.

Precision du Système			
PDF du corpus testé	Sections correctement récuperées	Sections détectées	Precision (en %)
Bourdin	2	3	66.6
Das	8	9	88.8
Gonzales	7	9	77.7
Iria	1	7	14.2
Mikheev	4	8	50
Mikolov	3	8	37.5
Nasr	2	8	25
Torres	6	8	75
Torres-Moreno	5	8	62.5

Notre programme a donc réalisé sur ce corpus test un traitement dont la précision moyenne d'élève à 55.2%.

4 Conclusion

Il est donc apparut, après conception et test de notre programme, que la fiabilité de celui-ci s'élève à 55.5%.
Ce résultat est critiquable et discutable sur plusieurs points.
En effet, nous avons pu noter au cours du développement du programme, au cours des nombreux test que nous avons effectués et des nombreux documents ainsi testés, que le manque de convention dans l'écriture des articles scientifiques, et les libertés parfois prise par les auteurs rendent le traitement des

textes problématiques.
En effet, les sections présentes, et parfois leur ordre, dans les documents à traiter varient d'un fichier à l'autre. Ainsi, la conception de parseurs parvenant à identifier les sections supposées présentes nécessite de prendre en considération ce critère, ce qui nous a valu de retravailler plusieurs fois notre façon de traiter les textes.
De plus, la precision obtenue correspond à l'évaluation de la fréquence à laquelle les sections des fichiers traités sont détectées et récupérées à la perfection. Ainsi, il suffit d'un charactère spécial non reconnu par python en début de section ou en fin, d'une prise de liberté par les auteurs du document, pour rendre le traitement d'un texte "imparfait" et ainsi diminuer la précision globale du système. Ainsi, nous avons pu observer qu'en dépit d'une precision paraissant moyenne à 55.2% le traitement des texte est dans l'ensemble réussi et les fichiers résultant entièrement exploitables par la suite, contenant toute l'information recherchée et nécessaire.