AVIGNON UNIVERSITÉ

Sprint 5

TD1 G01

Pierre De Saxce
Raphaël Rey
Guillaume Bonenfant
Sébastien Rocca
Nathanaël Lefèvre

10 décembre 2020

L3 Informatique
IL
UE Génie logiciel

Responsables
Juan Manuel Torres Moreno

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

Guillaume Bonenfant (guillaume.bonenfant@alumni.univ-avignon.fr)
Nathanaël Lefèvre (nathanael.lefevre@alumni.univ-avignon.fr)
Raphaël Rey (raphael.rey@alumni.univ-avignon.fr)
Sébastien Rocca (sebastien.rocca@alumni.univ-avignon.fr)
Pierre de Saxce (pierre.de-saxce-nahon@alumni.univ-avignon.fr)

1 Abstract

Dans cette étude, nous nous intéresserons aux performances d'un parseur de pdf que nous avons réalisé dans le cadre de l'Unité d'Enseignement "génie logiciel" dispensée par Avignon Université. Pour ce qui est du cadre de notre expérience, notre but était de récupérer des fichiers de sortie en .txt ou .xml contenant les informations principales d'articles scientifiques au format pdf. Pour ce faire, nous avons comparé deux système d'ORC (pdf2txt et pdftotext) pour sélectionner le plus fiable. par la suite, nous nous sommes basé sur les fichiers .txt généré par le système d'OCR pour les parser et récupérer le titre, l'abstract, les auteurs, l'introduction, le corps de l'article, la disscussion, la conclusion et les références bibliographiques. Nous nous intéresserons dans un premier temps à l'implémentation de notre parseur puis nous analyserons les performances de notre système.

2 Méthode

Pour parser des articles scientifiques en pdf vers une sortie en .txt ou .xml, nous nous appuyons sur la transcription des fichiers d'entrée en pure texte par le programme d'OCR pdftotext avec les options -nopgbrk et -raw, la première évitant les problèmes liées aux sauts de page et la seconde permettant de reconnaître les textes sur plusieurs colonnes.
Notre structure de récupération des données est basée sur un dictionnaire data associant les nom des différentes parties à récupérer à leur contenu détecté dans le fichier d'entrée. Pour ce qui est de la récupération des données à proprement parler, notre système se base sur une méthode itérative, ligne par ligne, et fonctionne grâce à des expressions Regex. Lorsqu'une partie est détectée, la variable "copy" prends la valeur de la clé correspondante dans le dictionnaire "data". tant que "copy" est égal au nom de la partie en cours et n'est pas vide, les lignes lues sont ajoutée à la valeur liée à la clé correspondant à "copy" dans data. Par exemple, si on détecte l'abstract, alors copy=="abstract" et les lignes lues sont enregistrées dans data["abstract"].

Par ailleurs, nous nous assurons de ne récupérer chaque partie qu'une seule fois. Ainsi, si plusieurs lignes matchent la règle de détection d'une partie, seule la première correspondance est prise en compte. Pour ce faire, on ajoute la clé de la partie en cours dans la liste "copied". On ne détecte le début d'une partie que si celle-ci n'est pas inscrite dans "copied". Le nom original du document est directement récupéré lors du choix du pdf à parser de façon triviale. Pour le titre du document, le système tente de le récupérer depuis les données au format doc du pdf parsé par le module pdfminer. Si cela ne fonctionne pas, on récupère la première ligne du document et nous considérons qu'il s'agit du titre de l'article. Les autres récupérations de données sont purement basées sur les regex et celles-ci ont un certain degré d'indulgence.

Voici, pour chaque partie, la règle appliquée pour détecter le début de la partie en question :

Auteurs : du début du document jusqu'au début de l'abstract.

Abstract : le mot "abstract" en début de ligne précédé de 0 à 5 caractères et suivi de 0 à 100 caractères puis d'un retour à la ligne. La casse est ignorée.

Introduction : le mot "introduction" en début de ligne précédé de 0 à 5 caractères et suivi de 0 à 3 caractères puis d'un retour à la ligne.La casse est ignorée.

Corps : de la fin de l'introduction jusqu'à ce qu'une autre partie soit détectée.

Discussion : le mot "Discussion" avec une majuscule ou tout en majuscule, précédé par 0 à 5 caractères et suivi par 0 à 40 caractères et un retour à la ligne.

Conclusion : Le mot "Conclusion" avec ou sans "s" ou "Summary" avec une majuscule ou tout en majuscule, précédé de 0 à 10 caractères et suivi de 0 à 40 caractères pour "Conclusion" et de 0 à 10 caractères pour "Summary". Le tout doit être suivi d'un retour à la ligne.

Références : Le mot "Reference" avec ou sans "s" avec une majuscule ou tout en majuscule, précédé par 0 à 5 caractères et suivi par 0 à 3 caractères et un retour à la ligne.

En règle générale, une partie se termine lors de la détection du début d'une partie suivante. Dans certains cas, une partie peut être terminée prématurément lors de la détection du numéro d'une nouvelle partie, notamment la fin de l'introduction lorsque l'on détecte une partie numérotée par un "2".

3 Résultats

PDF	Précision souple	Précision stricte
A Benders Decomposition Approach to Correlation Clustering	11/13	9/13
A memetic algorithm for community detectionin signed networks	9/14	9/14
An Improved Branch-and-Cut Code for the Maximum Balanced Subgraph of a Signed Graph	11/12	9/12
Cabrera RESUMES 2019	12/16	11/16
Conversational Networks for Automatic Online Moderation	12/14	11/14
Dynamical Models Explaining Social Balance and Evolution of Cooperation	7/14	5/14
Exact Clustering via Integer Programming and Maximum Satisfiability	12/14	12/14
LDA resume	12/16	9/16
Partitioning large signed two-mode networks/ Problems and prospects	11/16	11/16
Polibits 42 02	14/16	11/16
MOYENNE	76.8246337%	67.10164835%

4 Conclusion

En précision souple (+/-2 phrases de précision sur les frontières des parties), notre système réalise une performance de 76.8246337 % de réussite sur un corpus de 10 pdf mis à notre disposition pour évaluation. En précision stricte (à la phrase près), les performances baissent à 67.10164835% ce qui est très correct selon nous. Finalement nous pouvons considérer que le parsing d'articles scientifiques, si tant est que ceux-ci suivent un même modèle, n'est pas une tâche très difficile dans le monde de l'informatique. La difficulté emerge principalement du fait que tout le monde ne formate pas son document exactement de la même maniere. Si des étudiants de L3 sont capables d'obtenir un pourcentage de réussite supérieur à 70% en peu de temps, nous pouvons imaginer que des professionnels sont capables de faire beaucoup mieux. De plus l'appel à une IA permettrait d'obtenir des résultats encore meilleurs. Notons que sans IA, le parsing des articles ne doit pas pouvoir se faire très facilement avec un pourcentage de réussite vraiment élevé. Sans IA, si les articles scientifique suivaient une norme plus stricte, le parsing serait trivial. Cela reviendrait en effet presque à parser un document xml ou html.