Parseur d'articles scientifiques en format texte

Léo Palla
Nathan Flachaire
Ben Montheil

Université d'Avignon et des Pays de Vaucluse, 339 chemin des Meinajaries, 84140 Avignon, France

Abstract

Dans cet article, nous présentons un systèeme permettant de convertir des articles scientifiques en format pdf vers un format xml ou txt. Nous décrivons la méthode de fonctionnement de ce système et les résultats obtenus.

Keywords: Parseur, XML, PDF

1. Méthode

Pour parser les fichiers pdf, nous avons décidé d'utiliser python. Ce langage nous semblait adapté puisqu'il est facile à utiliser et est déjà maîtrisé par la totalité de l'équipe. Le programme va d'abord vérifier les arguments, à savoir le dossier contenant les fichiers pdf à analyser et le formattage désiré. Une fois ceci fait, une boucle se lance pour demander à l'utilisateur pour chaque pdf si il désire l'analyser ou non. Ensuite l'analyse et l'extraction de données peut commencer.

1.1. Récupération du texte depuis le pdf

Nous avions le choix entre deux outils pour convertir les fichiers pdf en fichiers textes, pdf2txt et pdftotext.
Nous avons porté notre choix sur pdftotext car il provoque moins d'erreurs d'encodage et qu'il crée automatiquement un fichier contenant le résultat de la conversion, contrairement à pdf2txt qui le renvoie sur la sortie standarde.

1.2. Analyse et traitement des données

Pour chaque fichier demandé à être analysé par l'utilisateur, le programme va rentrer dans une boucle. Premièrement, on crée une série de variable

booléenne qui seront utilisées pour délimiter si les différentes parties sont actuellement en train d'être lu ou pour savoir si les sections ont déjà été trouvées dans le texte. Une autre série de variable est initialisée, cette fois-ci ce sont des chaînes de caractères de base vides, servant de receptacle pour les différentes par parties.
On a donc une chaîne de caractère et au moins une variable booléenne pour chaque section que nous désirons parser.
A partir de la, le programme lance le script pdftotext sur le fichier pdf et ouvre le fichier texte correspondant pour commencer à le lire. Puisque nous lisons le fichier pdf dans l'ordre, il est nécessaire que les sections soient tout le temps ordonné de la même façon, ce qui est très souvent le cas (par exemple, le corps du texte ne sera jamais avant l'introduction). La première section demandée étant le nom du pdf original, nous n'allons pas la détailler puisque l'on récupère simplement le nom du fichier dans notre boucle.

1.2.1. Titre de l'article

La deuxième section à récupérer est donc le titre de l'article. Afin de la récupérer, nous sommes parti du postulat que le titre est forcément au tout début de l'article, sans rien avant. La variable booléenne du titre commence donc à True et on écrit le début de l'article jusqu'à tomber sur les auteurs.

1.2.2. Auteurs de l'article

Pour trouvers les auteurs, nous avons décidé d'utiliser le nom du fichier en partant du principe que au moins un des mots du nom du fichier est un nom d'un des auteurs. Ainsi le nom du fichier est découpé et chaque mot est stocké dans une liste. lorsque l'on rencontre un de ces mots pendant qu'on écrit le titre, on arrête d'écrire dans la chaîne correspondant au titre et on commence à écrire dans celle correspondant aux auteurs.

1.2.3. Abstract

Lorsque le mot "abstract" est rencontré pendant que les auteurs sont en trains d'être écrits, on passe les variables de l'abstract à True et celle des auteurs à False pour pouvoir écrire dans la chaîne correspondant à l'abstract.

1.2.4. Introduction de l'article

Cette fois ci, les éléments qui déclenchent le changement des variables sont par exemple le mot "Introduction" ou encore les élements de type "I." ou "1 ". Nous avons considéré la majorité des cas possibles afin de transitionner vers l'introduction au bon moment.

1.2.5. Corps de l'article

On commence à écrire dans le corps quand on a passé l'introduction, et donc qu'on arrive à la section 2 de l'article. C'est la partie la plus importante de l'article en termes de contenu, c'est donc aussi celle qui contient le plus d'erreurs. En effet, nous n'avions aucuns outils pour retranscrire les images ou les équations mathématiques avec les 2 convertisseurs proposé. Ces éléments sont donc souvent déstructuré ou rempacés par des erreurs d'encodage dans les fichiers formatés

1.2.6. Discussion

La discussion est un élément pas toujours présent dans les articles mais lorqu'il l'est il est toujours intitulé de la même façon, ce qui nous a permis de le récupérer sans trop de problème.

1.2.7. Conclusion

La conclusion est, contrairement à ce qu'on pourrait croire pas tout le temps présente, et des fois couplée ou remplacée par la discussion. Cependant quand elle est présente, c'est tout le temps dans une section nommée "Conclusion", donc peu de problèmes pour la récupérer aussi.

1.2.8. Bibliographie

Enfin, la dernière section à récupérer est la bibliographie, tout le temps intitulée "References" donc aussi facile à parser.

1.3. Mise en forme

Une fois que tout est récupéré, suivant l'argument passé par l'utilisateur, le programme va soit créer des fichiers textes, soit des fichiers xml, dans lequel il va écrire les châines trouvées et les mettre en forme.

2. Résultats

Nous réalisons une évaluation de la qualité du système en fonction de la précision obtenue sur un corpus de référence. La présence d'un fragment de texte extrait dans les références est comptabilisée positivement; son absence est pénalisée.
La précision est calculée comme suit :
Précision = Sections correctes trouvées par le système / Sections trouvées par le système

Pour Das_Martin, on obtient 8 sections au total dont 8 correctes. On a donc une précision de 1.
Pour Gonzalez_2018_Wisebe, on obtient 9 sections au total dont 9 correctes. On a donc une précision de 1.
Pour Iria_Juan-Manuel_Gerardo, on obtient 8 sections au total dont 8 correctes. On a donc une précision de 1.
Pour mikheev J02-3002, on obtient 8 sections au total dont 8 correctes. On a donc une précision de 1.
Pour Mikolov, on obtient 8 sections au total dont 8 correctes. On a donc une précision de 1.
Pour Torres-moreno1998, on obtient 8 sections au total dont 8 correctes. On a donc une précision de 1.
Cela fait donc pour le moment une moyenne de 1 si on considère uniquement les pdf bien construits.

Les 3 prochains pdf comportent des erreurs dans les fichiers de base au niveau du découpage en 2 colonnes qui les rendent difficiles à découper même à la main.
Pour Boudin-Torres-2006, on obtient 8 sections au total dont 3 correctes. On a donc une précision de 0.375.
Pour Nasr, on obtient 8 sections au total dont 6 correctes. On a donc une précision de 0.75.
Pour Torres, on obtient 8 sections au total dont 3 correctes. On a donc une précision de 0.375.
Cela fait donc une moyenne de 0.722, soit 72,2% de précision si on ajoute les pdf mal réalisés.

Le pdf jing-cutepaste est inutilisable, on ne le considère donc pas dans la précision du parseur.

3. Conclusion

Le parseur que nous avons réalisé est donc majoritairement fonctionnel, malgré certaines erreurs d'encodages ou facilités que nous nous sommes accordées.