Developpement d'un analyseur de texte d'articles scientifiques

Prénoms Noms

4 mai 2021

Résumé Les fichiers au format PDF, Portable Document Format, sont très appréciés, notamment pour leur portabilité comme leur nom l'indique, mais aussi pour leur présentation qui est généralement sobre et épurée. Malheureusement, ce format n'est pas des plus avantageux lorsqu'il s'agit d'en extraire des informations précises, notamment lorsqu'ils sont utilisés à des fins de recherches scientifiques. C'est pourquoi l'IRISA, l'Institut de Recherche en Informatique et Systèmes Aléatoires, met en compétition les étudiants de la promotion 2020/2021 en 3ème année de licence Informatique à Vannes afin qu'ils produisent un parseur d'articles scientifiques vers un format qui facilite l'utilisation d'un TAL, Traitement Automatique de la Langue. Ce document expose donc nos recherches et le développement mis en place pour répondre à cette demande.

1 Introduction

La première étape de ce travail a été de prendre en considération les outils qui nous étaient disponibles et de les évaluer afin de retenir ceux qui allaient nous permettre de réaliser notre analyseur.

Pour cela, nous avons eu le choix entre 2 convertisseurs de PDF vers texte, à savoir `pdftotext` et `pdf2txt`. En effet, cette partie est importante, car elle constitue le point de départ de notre travail puisque notre parseur se basera sur les résultats du convertisseur choisi pour faire son analyse. Ainsi, une partie de la qualité de notre travail va reposer sur l'outil choisi, si la retranscription du PDF vers un format texte est mal réalisée, c'est notre analyse qui sera faussée, c'est pourquoi leur comparaison est importante. La conclusion de cette étude nous amène donc à utiliser le package Linux pdftotext, notamment dans un souci d'optimisation et de rapidité. C'est donc depuis le résultat de cet outil que le développement de notre analyseur peut débuter.

2 Méthode

2.1 Outils annexes

Langage choisi D'un commun accord, dans le groupe, nous avons décidé de retenir le langage Python pour la facilité du langage, sa portabilité, sa rapidité d'exécution et son agilité de développement.

De plus, nous n'avons pas réalisé d'autres projets en Python, ce qui nous a grandement permis de nous améliorer.

Autres outils Par ailleurs, nous avons utlisé les outis `pdfinfo, pdftotext` et `pdf2txt`, pour transformer les fichiers PDF et en extraire les informations demandées.

Université de Bretagne Sud
E-mail:

2.2 Structure générale de notre programme

Notre programme s'appuie sur les phases suivantes :

1. parcours du répertoire
2. pour chaque fichier trouvé :
   — vérification si le fichier est un pdf
   — récupération du nom de fichier
   — récupération du titre
   — récupération du ou des auteurs, avec adresses mails et affiliations
   — récupération du résumé de l'article
   — récupération de l'introduction, du corps, de la conclusion et/ou discussion
   — et enfin les références bibliographiques
3. collecte des informations dans une liste de listes
4. écriture des fichiers résultats en XML et/ou TXT.

Nous utilisons en priorité `pdfinfo` qui extrait métadonnées du PDF. Enfin, une analyse sémantique est nécéssaire lorsque les métadonnées ne sont pas remplies.

Cette analyse sémantique est réalisée avec recherche de mots clés ou de certains motifs via des expressions régulières.

3 Résultats

3.1 Test de vitesse

À partir de la version 8 nous avons mis en place un test de vitesse pour vérifier la qualité de notre code. Nous avons testé notre code sur les instances du corpus, mais aussi sur des instances supplémentaires résumées dans le tableau 1.

3.1.1 Jeux de données

Pour effectuer nos tests de vitesse, nous disposons de répertoires tels que :

Corpus_2021 répertoire original de 12 fichiers pdf, totalisant 4.1Mo. Les métadonnées somt remplies pour 7 fichiers. Attention, parfois une métadonnée est incorrectement remplie, la métadonnée "Auteur" n'est pas l'auteur réel du papier.

Test-files 84 articles scientifiques (en pdf) publiés sur ScienceDirect, totalisant 210Mo. Les métadonnées des pdf sont correctement remplies.

Large-files 10 articles de grande taille, totalisant 168Mo, 13200 pages avec textes et images aléatoires. 5 articles ne comprenant aucune métadonnées (16Mo chacun), 5 articles avec métadonnées (18Mo chacun).

L'ensemble des données est disponible sur Github-Parser-pdf.

3.1.2 Méthodologie

Notre programme sera executé dix fois sur chacun des répertoires.

Pour calculer le temps moyen d'exécution du programme par fichier et par répertoire, nous allons utiliser la commande linux `time`.

Celle-ci nous donnera le temps total d'exécution du programme.

Nous diviserons le résultat obtenur par le nombre de fichiers, et nous pourrons obtenir un nombre qui correspondra à la vitesse moyenne d'exécution par fichier, et par répertoire en fonction du nombre de PDF.

Table 1 Résultats de l'exécution de la v8

Répertoire	Nombre de fichiers	Taille totale	Taille moyenne par fichier	Temps total*	Temps moyen par fichier*
Corpus_2021	12	4.2Mo	350Ko	24.031s	2.002s
Test-files	84	210Mo	2.3Mo	47.565s	0.566s
Large-files	10	168Mo	17Mo	17m51s	107.1s

(*) Temps moyen sur 10 exécutions

3.1.3 Présentation des résultats

Dans le tableau 1, vous trouverez les résultats des temps d'exécution du programme (v8) effectué par la procédure ci-dessus.

Dans le tableau 2, vous trouverez les résultats des temps d'exécution du programme (v8.1) effectué par la procédure ci-dessus.

Table 2 Résultats de l'exécution de la v8.1

Répertoire	Nombre de fichiers	Taille totale	Taille moyenne par fichier	Temps total*	Temps moyen par fichier*
Corpus_2021	12	4.2Mo	350Ko	6.513s	0.543s
Test-files	84	210Mo	2.3Mo	51.093s	0.608s
Large-files	10	168Mo	17Mo	9.187s	0.919s

(*) Temps moyen sur 10 exécutions

La figure 1 présente les temps d'exécution comparée des versions 8 et 8.1 par fichier.

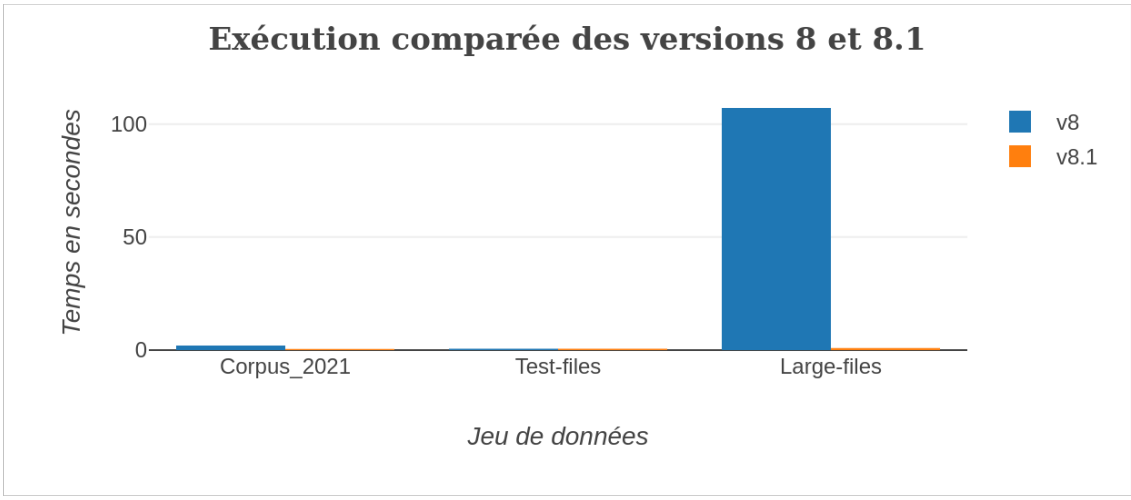

Figure 1 Temps d'exécution comparée des versions 8 et 8.1 par fichier

La différence entre les version 8 et 8.1 est que la version 8.1 n'utilise plus l'utilitaire `pdf2txt` qui traite intégralement un fichier même si on utilise l'option `-p 1` qui n'affiche que le résultat de la première page.

3.2 Résultats qualitatifs

Pour le sprint 5, correspondant à la version 21 de notre parser, nous devons maintenant faire une comparaison qualitative sur un nouveau corpus donné.

Pour comparer la qualité de nos fichiers analysés, il nous faut un corpus de référence que nous avons réalisé à la main.

Une fois celui-ci effectué, nous pouvons comparer les résultats de notre parser avec ce que nous devrions obtenir. Pour calculer la précision des résultats, nous comptons le nombres d'éléments trouvé par section, et nous comparons par rapport au fichier référent. Ensuite, en fonction du type d'évaluation, stricte (100% des mots sont exacts) ou souple (90% d'exactitude), nous calculons le pourcentage de sections retrouvées pour chaque fichier. Les sections à retrouver sont listées dans le paragraphe 2.2.

Le tableau 3 montre les résultats pour une précision souple :

Table 3 Résultat de la précision souple

Nom du fichier	(Sections, trouvées)	Précision
C14-1212.xml	(8, 6)	75.0 %
On_the_Morality_of_Artificial_Intelligence.xml	(5, 3)	60.0 %
b0e5c43edf116ce2909ae009cc27a1546f09.xml	(6, 4)	66.67 %
Guy.xml	(6, 4)	66.67 %
BLESS.xml	(5, 4)	80.0 %
infoEmbeddings.xml	(5, 3)	60.0 %
surveyTermExtraction.xml	(6, 1)	16.67 %
acl2012.xml	(8, 7)	87.5 %
L18-1504.xml	(7, 4)	57.14 %
IPM1481.xml	(8, 6)	75.0 %

Le tableau 4 montre les résultats pour une précision stricte :

Table 4 Résultat de la précision stricte

Nom du fichier	(Sections, trouvées)	Précision
C14-1212.xml	(8, 5)	62.5 %
On_the_Morality_of_Artificial_Intelligence.xml	(5, 2)	40.0 %
b0e5c43edf116ce2909ae009cc27a1546f09.xml	(6, 3)	50.0 %
Guy.xml	(6, 1)	16.67 %
BLESS.xml	(8, 2)	25.0 %
infoEmbeddings.xml	(6, 4)	66.67 %
surveyTermExtraction.xml	(5, 3)	60.0 %
acl2012.xml	(5, 4)	80.0 %
L18-1504.xml	(7, 1)	14.29 %
IPM1481.xml	(8, 2)	25.0 %

4 Conclusion

Malgré nos efforts (comme en témoigne le nombre élevé de versions notre parser), notre analyse durant ces sprints, nous permet de dire qu'il est impossible de trouver une règle générale pour extraire toutes les informations demandées dans tous les fichiers. En effet, il est toujours possible de construire un contre-exemple à une règle que nous aurions établie.

Le taux de réussite moyen en précision souple est de 64.465%, et de 44.013% en précision stricte.

Ces résultats sont encourageants et devraient inciter les chercheurs à remplir les métadonnées lors de l'écriture de leurs articles scientifiques. Cette action se fait simplement par l'ajout de la commande suivante en LaTeX :

```
\usepackage[pdftex,
            pdfauthor={ici il y avait nos noms},
            pdftitle={Développement d'un analyseur de texte d'articles scientifiques}
            ]{hyperref}
```

Ce projet nous a aussi permis de renforcer nos connaissances en Python, Git et Github, maîtrise des expressions régulières...