Projet de Développement

INF1603 réalisé

21 février 2021

Table des matières

Sprint 2 - Rapport

21 février 2021

1 Déroulement du programme

1.1 Langage choisi

D'un commun accord, dans le groupe, nous avons décidé de retenir le langage Python pour les raisons suivantes :

1. facilité du langage
2. portabilité du langage
3. rapidité d'exécution
4. agilité de développement

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

1.2 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

— récupération du résumé de l'article

3. collecte des informations dans une liste à double entrées
4. écriture des fichiers résultats

1.3 Spécificité des outils utilisés

Pour récupérer le titre et les auteurs, nous utilisons dans l'ordre les outils suivants :

1. `pdfinfo`
2. `pdftotext`
3. `pdf2txt`

Une analyse sémantique est nécéssaire lorsque les métadonnées ne sont pas remplies.

La conclusion(section 4) de ce document montre qu'il est impossible de trouver une règle générale qui permette de retrouver le titre ou les auteurs dans tous les cas.

En effet, il est toujours possible de construire un contre-exemple à une règle que nous aurons établie.

Si l'on se contente du répertoire Corpus_2021 comme champ d'application, cela est réducteur.

1.4 Dépôt Github

L'intégralité des codes Python et des jeux de données est disponible dès à présent sur Github-Parser-pdf dans le dossier `sprint2/parser-final/`.

Le parser final est la version **8.1** présentée et analysée dans ce document.

Pour utiliser nos programmes, il est impératif de lire le **README.md** sur Github.

2 Protocole de test de vitesse

2.1 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.

2.2 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.

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.

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

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

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

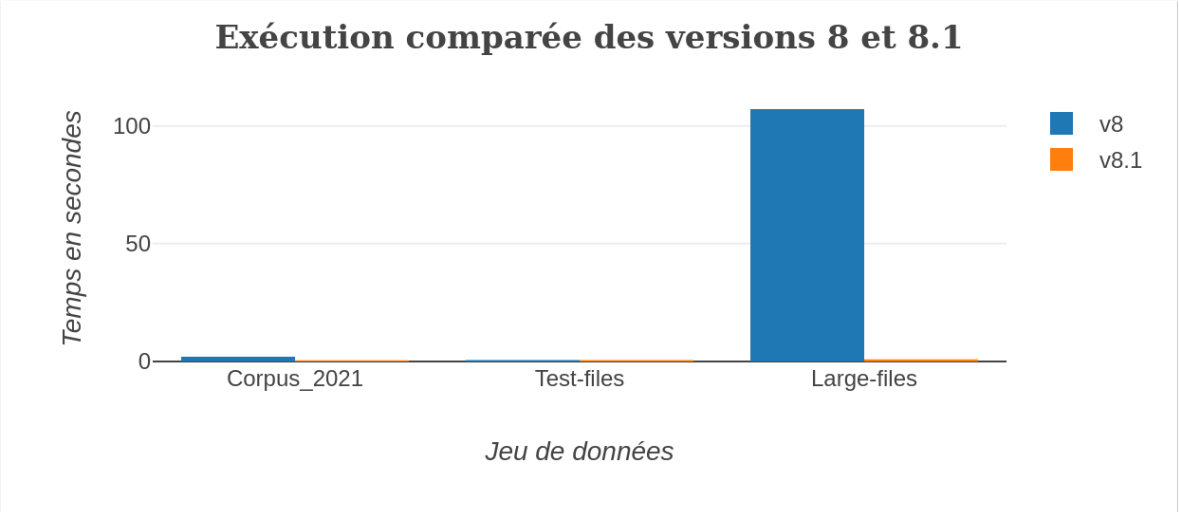

FIGURE 1 – Temps d'exécution comparée des versions 8 et 8.1 par fichier

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.

4 Commentaires sur les résultats obtenus

Comme vous avez pu le remarquer, les résultats obtenus dans les fichiers textes, sont variables. En fonction de comment sont créés les pdf, si les métadonnées sont remplies ou non, il peut être plus ou moins évident de récupérer ces données.

4.1 Particularitées

Auteurs : Vous avez pu voir sur certains PDF, le nom de Joe Pickert (par exemple) comme étant l'auteur de l'article. Cette information est-elle vraie ou fausse ?

Pourtant, quand on lit l'article, ce n'est pas ce nom qui apparaît, mais celui d'Andrei Mikheev. Alors, comment se fait-il que l'on obtienne 2 noms différents ?

Tout simplement, car l'utiliaire `pdfinfo` nous renvoit Joe Pickert comme auteur, mais, sauf à lire le pdf, il est impossible de savoir que ce n'est pas le bon auteur.

Il y aura toujours un cas (particulier ou non) que nous n'arriverons pas à gérer.

Ainsi, nous ne pouvons pas remettre en cause l'auteur trouvé dans les métadonnées de l'article.

Affichage en sortie : Comme vous pourrez le constater, dans certaines sorties texte, il existe une ligne blanche supplémentaire à la fin du titre.

Cette erreur d'affichage ne peut être résolue qu'avec un traitement particulier pas toujours possible. Il existe dans certains titre un caracètre de code ASCII 06A (lf) que la fonction `replace` de Python ne peut pas traiter.

Nous avons donc choisi de laisser ces "erreurs" d'affichage pour éviter des traitements qui pourraient nuire au contenu des titres non concernés.

4.2 Recommendations

Pour générer correctement les métadonnées dans les fichiers PDF, nous recommendons fortement d'utiliser la commande suivante dans la génération de fichier PDF à partir de code LaTeX :

```
\usepackage[pdfusetitle]{hyperref}
```