Parseur de PDFs

Antoine Jamelot, Université de Bretagne Sud
jamelot.e1500523@etud.univ-ubs.fr

Sofiane Ben Massaoud, Université de Bretagne Sud
ben-massaoud.e1803113@etud.univ-ubs.fr

Baptiste Colas, Université de Bretagne Sud
colas.e1800392@etud.univ-ubs.fr

Théau Huteau, Université de Bretagne Sud
huteau.e1803128@etud.univ-ubs.fr

3 mai 2021

1 Définition

Parser : (Anglicisme informatique) (Programmation) Parcourir le contenu d'un texte ou d'un fichier en l'analysant pour vérifier sa syntaxe ou en extraire des éléments (définition du Wictionnaire).

2 Abstract

Cet article a pour but de présenter notre production dans l'Unité d'Enseignement : Projet de Développement. Nous vous présenterons les méthodes utilisés et les raisons de ses choix, ainsi que nos résultats finaux sur un corpus de référence et une discussion sur ses résultats. Nous avons donc parsé les articles du corpus tout d'abord en convertisant l'article au format txt ou xml selon la commande rentrée en ligne de commande. Ensuite on normalise le texte, c'est à dire qu'on le transforme en un texte d'une seule colonne et on procède à un parsage par section en cherchant des chaînes de caractères représentant le titre de la section recherché. Une fois les articles parsé, on compare un corpus d'article parsé à la main à un corpus parsé avec l'outil que nous avons produit à l'aide de la fonction précision.py. Par la suite nous observons les résultats obtenu.

3 Méthode

3.1 Conversion des *PDF* au format *TXT*

Pour parser ce corpus de référence, l'une des premières étapes est de convertir les articles initiallement au format PDF au format TXT. Pour cela nous avons pris le temps de choisir entre plusieurs modules. Notre recherche nous a amenené à choisir pdftotext car il permettait de conserver le format des articles comportants plusieurs colonnes et au vue des différents tests que nous avons effectués avec différents modules du genre pdftotext semblait le plus prometteur (voir rapport sur GitHub). Mais pdftotext avait ses désavantages, par exemple pour certains titres de section une lettrine venait s'insérer lors de la conversion, "REFERENCES" devenait "R EFERENCES", ce qui pouvait poser des difficultés par la suite dans la délimitation des différentes sections. A noté que nous avons aussi utilisé PyPDF2 car il présente des fonctionnalités intéressantes sur la manipulation des métadonnées des articles comme les auteurs, le titre, etc...

3.2 Conversion des *PDF* au format *XML*

Par la suite, il nous a été demandé de convertir les PDFs en XML. Le module initialement choisi (pdftotext) permetait cela. Ce qui a donc renforcé notre choix de nous vers pdftotext pour la conversion des PDFs.

3.3 Choix du langage

Pour réaliser ce projet nous avons choisi le langage python. Nous avons fait ce choix pour plusieurs raisons que je vais détailler ici.
Python a l'avantage d'être a comprendre et à apprendre. Il est aussi un langage à usage général. Ce langage comporte l'un des gestionnaire de paquet les plus matures, gestionnaire de paquet qui comporte pdftotext. Mais surtout dans notre situation, Python avait l'avantage d'être connu et maîtriser dans son ensemble par tout les membres du groupes. De plus, c'est un langage bien documenté.

3.4 Normalisation du texte

Les articles du corpus sur lesquelles nous avons travailler tout le long du projet étaient souvent des articles comportant plusieurs colonnes et cela peut poser problème. En effet, car lorsque vous lisez une ligne de manière classique (avec les fonctions *readlines()* ou *read()* en python) le texte n'est pas lu une colonne après l'autre mais chaque ligne de chaque colonne en même temps, donc il faudrait soit lire colonne par colonne soit normaliser les textes comportants plusieurs colonnes. Etant donné que certains textes ne contiennent pas plusieurs colonnes, nous avions à l'esprit d'opté pour la deuxième option car nous voulions aussi enlever du parsage les en-têtes et pieds de pages qui pouvaient poser problème par la suite.
M. Jamelot a donc implémenté une fonction qui avait pour but de normaliser les textes dans la classe page.py.

3.5 Parsage par section

Après avoir normaliser l'ensembles des articles, nous nous sommes penchés aux parsages des articles section par section. Pour parser une section, au départ nous cherchions dans le texte une chaîne de caractères (une string en Python) qui correspondait au titre de la section recherché. Lorsque la chaîne était trouvé, nous avions donc reperé ce qui était supposé être le début. Ensuite nous cherchions le début de la section suivante et nous avions notre deuxième "flag"/repère. Alors à ce moment là, en mémorisant le numéro de ligne et de page du début et de la fin de la section. Nous retournions un champ avec le numéro de page de ligne du début et de la fin. Ensuite, nous écrivions ce texte dans un txt ou xml selon le paramètre passer en ligne de commande.
Mais, les titres comportaient des fautes du à la conversion. Donc, si nous cherchions une chaîne de caractères, cela pouvait être non fructueux. Il fallait donc effectué une recherche de plusieurs chaînes de caractères et chercher un mot seul sur sa ligne. Car après un titre de section, il faut aller à la ligne.

3.6 Exploitation des métadonnées

Les métadonnées, lorsqu'elles étaient présentes ont permis d'améliorer la précision du module car, elles ont permis de déterminer avec exactitude le titre par exemple ou certains auteurs (mais pas leurs affiliations).

3.7 Menu graphique tkinter

Il nous a été aussi demander de permettre de permettre le parsage par un "menu graphique". Ce que j'ai donc permis, en lancant comme décris dans la section du README : "Lancement de l'outil"-"Parsage", paragraphe "Menu tkinter".

4 Résultats

Nous allons maintenant nous interesser aux résultats que nous avons avec l'outil dans sa dernière version, c'est à dire sa version dans le sprint 4. Nous verrons comment fonctionne l'outil sur un corpus de référence composé de 10 articles et nous nous interresserons à sa précision. Cette précision est calculé en comparant un article parsé "à la main" et un article avec l'outil. Ensuite on compare les deux parsages sections par sections, c'est à dire le titre les auteurs un à un, l'abstract, l'introduction, la conclusion, les résultats et les références.
Cette précision est calculé comme ceci :
Précision = (Titre+Auteurs+sections correctes)/(Titre+Auteurs+sections véritables)
Et on obtient un chiffre inférieur ou égal à un que l'on peut convertir en pourcent.
Ce chiffre correspond donc à la précision du parsage de notre système sur le corpus de référence. Nous allons donc observé ces résultats.

4.1 Présentation résultats

Article	Précision stricte (de 0 à 1)	Précision souple (de 0 à 1)
BLESS.pdf	0.5	0.875
C14-1212.pdf	0.375	0.75
IPM1481.pdf	0.125	0.375
b0e5c43edf116ce2909ae009cc27a1546f09.pdf	0.375	0.5
infoEmbeddings.pdf	0.375	0.375
surveyTermExtraction.pdf	0.25	0.5
$\mathrm{On}_t he_M orality_o f_A rtificial_I ntelligence.pdf$	0.334	0.334
L18-1504.pdf	0.625	0.625
Guy.pdf	0.858	0.858
acl2012.pdf	0.125	0.625
Moyenne	**0.394**	**0.582**

TABLE 1 – Résultats du parsage des 10 articles du corpus de référence.

4.2 Discussion des résultats

On peut voir que les résultats sont loins d'être parfaits. Mais, cependant, l'ensemble de l'équipe s'y attendait. Car, nous avions remarqué que les auteurs et leurs affiliations n'étaient jamais écris d'une même manière ce qui a rendu le parsage des auteurs et leurs affiliations assez difficile. Donc pour chaque auteurs incorrectes cela diminuait la précision d'environ 17% voir plus (car un article comporte 6 sections ou plus et donc 1 sur 6 vaut environ 16,666...%). Mais pourquoi ces difficultés avec les auteurs et affiliations ? Cela est du à une non-unifmromisation de l'écriture des auteurs et leurs affiliations à travers les différents articles du corpus, d'où les résultats de globalement 48,8%.

5 Conclusion

Nous vous avons donc présenté notre outil de parsage. Il est certes imparfait, nous aurions pu produire un outil qui donnait une précision de 100% sur certains articles mais ce n'était pas l'objectif du projet. L'objectif du projet était de faire un outil parsant des articles scientifiques même si leur structures différait. C'est donc ce que nous avons essayé de faire au mieux.

6 Références

Python 3.9.4 documentation : Lien vers documentation