Development of a Python Parser to Extract Information from a Scientific Paper in PDF format and write an output file

Clément Le Bihannic-Bertôt
Université Bretagne Sud 56000 Vannes, France
le-bihannic-bertot.e2000371@etud.univ-ubs.fr

Clément Cano
Université Bretagne Sud 56000 Vannes, France
cano.e2000507@etud.univ-ubs.fr

Théo Genette
Université Bretagne Sud 56000 Vannes, France
genette.e2001218@etud.univ-ubs.fr

Evann Perrinel
Université Bretagne Sud 56000 Vannes, France
perrinel.e2000399@etud.univ-ubs.fr

Iwan Fouquet
Université Bretagne Sud 56000 Vannes, France
fouquet.e2002491@etud.univ-ubs.fr

May 2023

Abstract

L'IRISA avait besoin d'un analyseur de documents texte générés à partir d'un PDF. L'objectif était de découper le mieux possible les sections d'un article scientifique afin que les chercheurs gagnent du temps dans la lecture des articles. L'IRISA souhaitait un système écrit dans un langage de programmation adéquat et fonctionnant sur un système GNU/Linux. Pour répondre à ce besoin nous avons formé une équipe de 5 étudiants et travaillé en mode agile avec la méthode SCRUM.

1 Introduction

Notre objectif était de convertir un fichier PDF vers un fichier au format XML ou TXT. Pour cela, nous avons décidé que la manière la plus simple et la plus rapide de mettre en place notre projet était d'utiliser le langage Python. À la fois pour sa flexibilité et pour l'intégration très simple des expressions régulières dans ce langage, élément central dans notre projet. En plus d'une version utilisable via un terminale, nous disposons d'une interface graphique bien plus simple à utilisée et toujours réalisée en Python à l'aide du module TKinter et CustomTKinter. TKinter est la librairie graphique de base du langage Python, elle ne permet de faire que des interfaces très simples mais c'était pour nous largement suffisante puisque nous devions sélectionner un fichier sur le système, choisir entre deux radio boutons, et sauvegarder un fichier. CustomTKinter nous a servi à rendre plus agréable visuellement l'interface créée à l'aide de TKinter. Avec cette librairie il a été par exemple possible de rendre l'interface plus sombre afin de surligner en couleurs les aspects importants sans pour autant perdre en esthétisme, et de manière générale, de produire une interface moderne et lisible.

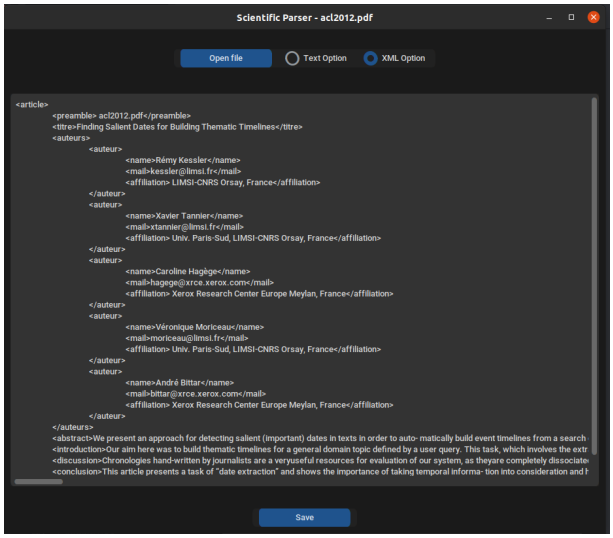

Figure 1: Interface Graphique de l'application

2 Explication du système

Concernant le code écrit dans ce projet, dee manière générale, nous avons éllaboré une expression régulière pour chaque partie présente dans tout article scientifique (abstract, results, conclusion, ...). Cette dernière était plus ou moins complexe selon les parties afin de capturer le texte concerné. Prenons par exemple la partie Abstract. Nous avons tout d'abord écrit une expression régulière comportant des groupes nommés pour retrouver simplement l'information qui nous intéressait mais aussi de vérifier la présence de certains éléments comme un point à la suite du numéro de la section. Pour simplifier le débug de notre expression régulière nous avons ajouté dans chaque méthode une boucle for permettant d'afficher chaque groupe capturé et le texte associé. Enfin, nous avons récupéré les index des groupes qui nous intéressaient (souvent le texte uniquement) et nous avons appliqué de nouveau l'expression régulière sur tout le texte.

```python
def getAbstract(text,file_name=""):
    """
    Extracts the abstract from the PDF using multiple regex to achieve the highest accurency
    possible
    """
    if(file_name=="IPM1481.pdf"):
        regex=r"(?:([1-9]+?.?)|([IVX]*.))?(\s+)?(?:(Abstract)|(ABSTRACT)|abstract)((\s+)?   *\
    n?)(?P<text>(?:.|\n)*?)(^((([1-9]+?.?)|([IVX]*.))?\s+?(Introductio.n|INTRODUCTION|
     A r t i cial ))|  ntroductio  .n\n)"
    else:
        regex = r"(?:([1-9]+?.?)|([IVX]*.))?(\s+)?(?:(Abstract)|(ABSTRACT))((\s+)?   *\n?)(?P<
    text>(?:.|\n)*?)(^((([1-9]+?.?)|([IVX]*.))?\s+?(Introductio.n|INTRODUCTIO.N| A r t i cial ))|
    Introductio.n\n)"
    matches = re.finditer(regex, text, re.MULTILINE)
    # Parcours de tous les groupes pour debug
    for matchNum, match in enumerate(matches, start=1):

        #print ("Match {matchNum} was found at {start}-{end}: {match}".format(matchNum =
    matchNum, start = match.start(), end = match.end(), match = match.group()))

        for groupNum in range(0, len(match.groups())):
            groupNum = groupNum + 1
            #print ("Group {groupNum} found at {start}-{end}: {group}".format(groupNum =
    groupNum, start = match.start(groupNum), end = match.end(groupNum), group = match.group(
    groupNum)))


    # Rechercher le texte correspondant a la regex
    groups = re.compile(regex)
    textIndex = groups.groupindex['text']
    abstract_match = re.findall(regex, text,re.MULTILINE)
    abstract = "N/A"
    # Verifier si un resultat a ete trouve
    if abstract_match:
        # Afficher le texte extrait
        abstract = match.group(textIndex)
    return abstract.replace('\n',' ')
```

Listing 1: fonction getAbstract()

Une fois l'ensemble de nos expressions régulières écrites pour toutes les parties d'une publication, il ne restait plus qu'à écrire une méthode permettant de générer le fichier xml ou le fichier texte. Cette étape était simple, il suffisait d'écrire dans un fichier les chaines de caractères récupérées dans nos méthodes, au détail près qu'il fallait rajouter les balises XML dans le cas du fichier XML en sortie.

```python
def writeXML(file_name,output_file_name,text,metadata,pdf):
    """
    Writes all the capital information in a .xml file with an XML layout
    """
    outputXML = "<article>\n"
    outputXML+="\t<preamble> "+file_name+"</preamble>\n"
    outputXML+="\t<titre>"+getTitle(metadata,text)+"</titre>\n"
    outputXML+="\t<auteurs>\n"
    auteursInfo = getAuthors(metadata,text,getTitle(metadata,text))

    for i in auteursInfo:

        outputXML+="\t\t<auteur>\n"
        outputXML+="\t\t\t<name>"+i+"</name>\n"
        outputXML+="\t\t\t<mail>"
        outputXML+=auteursInfo[i]['mail']
        outputXML+="</mail>\n"
        outputXML+="\t\t\t<affiliation>"+auteursInfo[i]['affiliation']+"</affiliation>\n"
        outputXML+="\t\t</auteur>\n"
    outputXML+="\t</auteurs>\n"
    if(file_name=="IPM1481.pdf"):
        outputXML+="\t<abstract>"+getAbstract(pdf.pages[1].extract_text(),file_name)+"</
    abstract>\n"
```

```
    else:
        outputXML+="\t<abstract>"+getAbstract(text)+"</abstract>\n"
    outputXML+="\t<introduction>"+getIntroduction(text,file_name)+"</introduction>\n"
    outputXML+="\t<discussion>"+getDiscussion(text)+"</discussion>\n"
    outputXML+="\t<conclusion>"+getConclusion(text,file_name)+"</conclusion>\n"

    outputXML+="\t<biblio>"+getBiblio(text,file_name)+"</biblio>\n"

    outputXML+= "</article>"
    if(output_file_name!=""):
        fd = os.open(output_file_name,flags=os.O_RDWR |os.O_CREAT | os.O_TRUNC)
        text = str.encode(outputXML)
        lgtext = os.write(fd,text)
        if(lgtext==0):
            sys.stderr("Aucune donnees n'a pu etre extraite")
        os.close(fd)
    return outputXML
```

Listing 2: fonction writeXML()

3 Résultats

Plus le projet avançait plus les résultats que nous avons obtenus s'amélioraient. Nous vous présenterons donc ici les résultats obtenus pour le dernier sprint du projet basés sur un corpus d'articles scientifiques de test. Voici les résultats obtenus sur ce dernier :

	Moyenne		acl2012	b0e5c43edf116c	BLESS	C14-1212	Guy	infoEmbeddings	IPM1481	L18	AI-Morality	SurveyTerm
Preamble	99,333	=	100	100	100	100	93,33	100	100	100	100	100
Titre	99,217	=	99,07	99,29	98,99	98,99	99,47	100	100	100	100	96,36
Auteurs	88,011	↗	99,4	93,37	93,4	96,27	99,56	92,95	94	90,43	66,05	54,68
Introduction	92,767	↗	99,85	98,85	99,78	93,9	99,87	96,72	59,99	99,93	100	78,78
Abstract	95,026	↗	99,68	99,66	99,67	99,7	99,86	82,59	99,2	99,64	99,01	71,25
Conclusion	88,298	↗	98,89	89,31	82,28	96,48	99,14	98,29	49,87	98,82	96,72	73,18
Bibliographie	89,766	↗	95,78	98,43	97,56	96,15	65,38	93,55	93,05	71,74	96,29	89,73
Total	93,081	↗	97,73	96,99	95,95	97,36	93,8	94,87	85,16	94,37	94,01	80,57
			=	↗	↗	↘	↗	↗	↗	↗	=	=

Figure 2: Résultat finaux du logiciel

Nous pouvons observer que les cellules en vert sont les meilleurs résultats alors que ceux en blancs et en rouges sont moins bons. Ces scores décevants peuvent s'expliquer de plusieurs manières. Premièrement nos expressions régulières peuvent ne pas fonctionner en toutes circonstances et donc ne pas capturer complètement les parties de texte concernées. On a également retrouvé des différences notables entre les documents de références permettant d'obtenir les scores et le contenu véritable des PDF qui devait être extrait, provoquant parfois de légères baisses au score final. Durant ce sprint, nos scores n'ont pas cessés d'augmenter, en grande partie en raison de l' amélioration de notre recherche des auteurs qui n'était jusqu'alors pas ou peu intégrée au code, et ce jusqu'à atteindre un score de 93%.

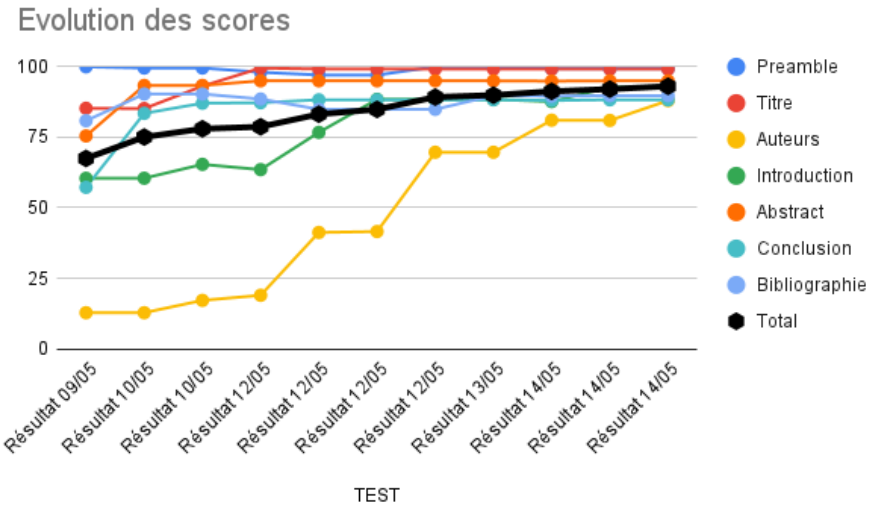

Figure 3: Evolution des scores

4 Conclusion

Notre système a su répondre aux attentes de notre client au vu des bons résultats que nous avons obtenus. Grâce à des ajustements durant le sprint final sur le corpus de test, nous avons optimisé le plus possible l'ensemble de nos expressions régulières afin de capturer un maximum d'information, et d'augmenter le taux de précision global au meilleur niveau possible. Ce cadre de la méthode agile SCRUM nous a permis lors de chaque sprint d'améliorer un peu plus notre méthode ainsi que nos scores sur l'ensemble des textes. Le projet nous a fait découvrir une nouvelle façon d'appréhender le travail de groupe. Cependant nous avons tout de même rencontré quelques difficultés tout au long du projet :

- Le changement entre le corpus d'entraînement et celui de test qui comportait des PDF formattés de nouvelles manières
- Le contenu incorrect et très changeant des XML utilisé pour calculer le pourcentage qui nous ont fait changer très souvent nos méthodes à la fin du projet
- L'outil utilisé pour convertir nos PDF en texte qui n'était pas toujours très performant et introduisait des erreurs ou complications dans le texte à analyser

Le système produit à l'issu de ce projet est complet, efficace avec un score largement satisfaisant et ne contient que très peu de règles mises en place uniquement pour un fichier. On peut donc penser qu'il garderait une efficacité très suffisante en cas de généralisation et d'utilisation sur de nouveaux articles correctement formattés.