11.18.05\_RAP\_V01-BD Trappes, 20/07/2018

OTP : 1.P.DOC0001

**INTElligence Artificielle POUR DOCUMENTS SCIENTIFIQUES**

**programme de reconnaissance automatique de documents scientifiques DANS LE RESEAU**

**ETUDE POC (proof of concept) avec PYTHON**

|  |  |  |
| --- | --- | --- |
| **DIFFUSION :** | ***ERAMET IDEAS*** |  |
|  | LAURENT JONCOURT  JEAN-PIERRE CESCUTTI |  |
|  | YVES LE QUESNE  SARAH BELAIR |  |
|  |  |  |

|  |  |
| --- | --- |
| KEYWORDS **:** | KNOWLEDGE MANAGEMENT ° INDUSTRY 4.0 ° PYTHON ° MACHINE learning ° RESEARCH ° ERAMET IDEAS ° OPPORTUNITY |

|  |  |
| --- | --- |
| Auteurs : T. BUI B. DUSANTER | Correcteur : A. DIANOUX |
|  |  |

|  |  |
| --- | --- |
| Approbateur : B.LABOUDIGUE | Emetteur : M. LENOIR |
|  |  |

**RESUME :**

|  |
| --- |
| La collecte du savoir scientifique est réalisée à ERAMET Research depuis la fondation du centre en 1972. La dotation documentaire initiale provint du fonds des rapports scientifiques et collections de revues scientifiques de Minerais & Métaux, et une grande opération d’achat de livres eut lieu vers 1972, alors que le département comptait 9 personnes.  Cette collecte de documents papier fut poursuivie dans les années 80 et 90, dans des machines dites « Kardex » dans lesquelles on collectait, dans des dossiers thématiques, les documents papiers internes et externes. Vers la fin des années 90, l’apparition de la technologie du scan invita à basculer dans l’achat d’une GED[[1]](#footnote-1), qui existe encore sous le nom CINDOC. Le patrimoine des rapports internes qui concernaient le nickel fut scanné pour un client notable : la SLN.  En 2009, le logiciel de recherche Polyspot fut acheté pour permettre de faciliter les recherches de documents scientifiques depuis un certain nombre de filiales choisies, et assurer l’envoi de ces documents par e-mail. Aujourd’hui, par le libre accès à internet, on estime que les chercheurs recueillent pour moitié leurs bibliographies à partir de la base interne, et pour moitié à partir de leurs relations, leurs collections personnelles ou internet. Il y a de grosses disparités individuelles.  En 2016, la mini-T opéra un changement important dans les us-et-coutumes, par la mise en place d’un serveur T:\PROJET qui permet désormais à tout membre d’une équipe projet, quelle que soit sa spécialité ou entité, de mettre au même endroit partagé les documents pertinents d’un projet, y compris les documents bibliographiques. La grande versatilité de la disponibilité des documents internet a poussé, lors de la mini-T, à écrire une procédure demandant aux chercheurs de mettre leurs documents bibliographiques dans ces répertoires, et ainsi d’assurer leur préservation.  L’activité documentaire doit encore gagner en productivité, et la présente étude a pour ambition de tester, sur un POC[[2]](#footnote-2), la faisabilité d’une collecte automatisée des documents scientifiques du réseau PROJET, à partir d’une reconnaissance automatisée du caractère « scientifique » de ces documents.  Réalisé en code Python, par deux méthodes testées en parallèle, la classification est possible : **la conclusion du POC est clairement positive**. Une technique d’apprentissage automatique correctement programmée permet de réaliser un tel travail, à partir d’analyse du texte, et vraisemblablement aussi à partir de l’analyse des images.  Néanmoins, un POC n’est pas une solution pérenne : il faut encore que plusieurs aspects, normalement résolubles, soient traités à leur tour :   1. Le cas échéant, développer un programme Python pour OCRiser[[3]](#footnote-3) ces fichiers et récupérer le texte et l’image de la première page 2. Intégrer d’éventuels critères d’analyse graphique pour améliorer le pouvoir discriminant du logiciel d’I.A. 3. Dédoublonner les fichiers recueillis de la base KARDEX+BIB, par réalisation d’une connexion de Python avec la base KARDEX avec CINDOC. 4. Mettre ce programme à disposition de l’indexeur Polyspot |

**HISTORIQUE DES MODIFICATIONS**

|  |  |  |
| --- | --- | --- |
| Version | Date | Objet |
| 01 | 20/07/18 | Création |

**TABLE DES MATIERES**

[1. Introduction 4](#_Toc519856927)

[1.1. Rappel des principes du « machine learning » 4](#_Toc519856928)

[1.2. Classification des documents 5](#_Toc519856929)

[1.3. Pourquoi le choix de Python ? 5](#_Toc519856930)

[2. Etude sur des fichiers pdf avec texte disponible 5](#_Toc519856931)

[2.1. Collecte de données (collecte de documents) 5](#_Toc519856932)

[2.1.1. Les documents dits ‘SciDoc’ 5](#_Toc519856933)

[2.1.2. Les documents dits ‘NonSciDoc’ 6](#_Toc519856934)

[2.2. Calcul des traits caractéristiques (features) sur base .TXT 6](#_Toc519856935)

[2.3. Developpment et evaluation de Modèles de « machine learning » 7](#_Toc519856936)

[2.3.1. Evaluation avec l’algorithme *naïve bayes* 7](#_Toc519856937)

[2.3.2. Evaluation avec des modèles d’apprentissage automatique avancés 7](#_Toc519856938)

[3. Etude COMPLEMENTAIRE sur des pdf avec texte 8](#_Toc519856939)

[3.1. Collecte de données (collecte de documents) 8](#_Toc519856940)

[3.2. Extraction de caractéristiques 8](#_Toc519856941)

[3.2.1. Lire le contenu du fichier pdf 8](#_Toc519856942)

[3.2.2. Extraction de carateristiques 8](#_Toc519856943)

[3.3. Developpment et evaluation de Modèles de « machine learning » 9](#_Toc519856944)

[4. Perspectives d’utilisation 10](#_Toc519856945)

[4.1. Documents du réseau Eramet Research 10](#_Toc519856946)

[4.2. Documents du réseau Eramet Ingénierie 10](#_Toc519856947)

[4.3. Quelques risques néanmoins repérés 11](#_Toc519856948)

[4.4. Calcul des traits caractéristiques a partir d’images 11](#_Toc519856949)

[5. Prochaines étapes. Pré-ConcLusion 12](#_Toc519856950)

[6. Autres applications possibles 12](#_Toc519856951)

[7. Bibliographie 12](#_Toc519856952)

[8. Annexe I 13](#_Toc519856953)

[8.1. Calcul de la liste des fichiers a lire 13](#_Toc519856954)

[8.2. Lire le contenu du fichier pdf 13](#_Toc519856955)

[8.3. Extraction des features pour document txt 13](#_Toc519856956)

[8.4. Extraction des features pour des documents pdf natif 14](#_Toc519856957)

[8.5. « Grid search » et « cross validation » pour trouver des parametres optimisés 15](#_Toc519856958)

[8.6. Evaluation des modeles d’apprentissage automatique 16](#_Toc519856959)

# Introduction

## Rappel des principes du « machine learning »

Selon la définition d’Arthur Samuel (1959), l’apprentissage automatique[[4]](#footnote-4) est la discipline donnant aux ordinateurs la capacité d’apprendre sans qu’ils soient explicitement programmés. Donc, l’apprentissage automatique est la science de programmer les ordinateurs de sorte qu’ils puissent « apprendre » à partir de données. Les machines apprenantes servent à classifier des informations en découvrant des structures d’organisation interne dans les données.

Les systèmes de l’apprentissage automatique peuvent être classés en fonction de la quantité et du type de supervision qu’ils reçoivent pendant l’entraînement. Il existe quatre grandes catégories : l’apprentissage supervisé, l’apprentissage non-supervisé, l’apprentissage semi-supervisé et l’apprentissage par renforcement. Dans l’apprentissage supervisé, les données d’entraînement que nous transmettons à l’algorithme comprennent les solutions souhaitées, appelées étiquettes (labels en anglais). Une tâche d’apprentissage supervisé typique est la classification qui s’appuie sur un corpus d’entraînement contenant l’étiquette correcte pour chaque entrée.

Le schéma utilisé par la classification, qui consiste en deux phases : entraînement et prédiction, est illustré à la Figure 1. Pendant l’entraînement, un extracteur de caractéristiques est développé pour convertir chaque valeur d’entrée en un jeu de caractéristiques (*feature set*). Ces jeux de caractéristiques saisissent l’information de base sur chaque entrée qui devrait être utilisée pour la classer. Des paires de jeux de caractéristiques et d’étiquettes sont introduites dans l’algorithme d’apprentissage automatique pour générer un modèle prédictif. Pendant la prédiction, le même extracteur de caractéristiques est utilisé pour convertir les entrées inconnues en jeux de caractéristiques. Ces ensembles de caractéristiques sont ensuite introduits dans le modèle, qui génère des étiquettes prédictées.

Input

Machine learning algorithm

Label

(a) Training phase

Trained classifier model

Feature extractor

Label

Features

Input

Feature extractor

Features

(b) Prediction phase

Figure 1. Schéma de principe d’une « machine learning » à classification supervisée

L'extraction de caractéristiques discriminantes (ingénierie des caractéristiques ou « feature engineering ») a besoin d’un grand effort et des expertises dans le domaine. Dans les techniques d'apprentissage profond[[5]](#footnote-5), l'étape d'ingénierie des caractéristiques peut être automatisée par des algorithmes, à condition qu'il y ait suffisamment de données et de puissance de calcul.

## Classification des documents

Le développement du programme de classification des documents est basé sur le schéma présenté à la Figure 1. Il se compose de trois taches : collecter des données (i.e. documents), développer l’extracteur de caractéristiques et mettre en œuvre les algorithmes d’apprentissage automatique.

Les matériaux à classifier sont des documents pdf. En effet, tous les acteurs du marché délivrent des documents scientifiques sous ce format. Le format antérieur, dit « tif » est devenu obsolète, si bien qu’un document « tif » du réseau est forcément tiré de notre base documentaire, et n’a donc pas besoin d’être collecté.

Selon leur origine, dans les divers systèmes informatique d’ERAMET, on trouve plusieurs sortes de **documents pdf**

1. des scans à l’état pur : (il n’y a que des images)
2. des scans contenant du texte OCRisé en surimpression transparente (on le teste en vérifiant que le texte est sélectionnable avec le curseur)
3. des documents natifs à texte non protégé (fabriqués par MS-Word, pour PowerPoint)
4. des documents natifs à texte protégé (l’auteur souhaite empêcher les indexeurs de copier le texte)

Dans ce POC, nous présentons les méthodes de classification des trois premiers types de documents pdf ; à noter que nous utilisons actuellement un logiciel de traitement OCR semi-automatisé, qui ramène le cas (a) au cas (b).

## Pourquoi le choix de Python ?

Python est un langage de programmation orienté à la fois objet et dont la syntaxe s’approche assez bien du formalisme des mathématiques dites modernes (théorie des ensembles). De nombreuses bibliothèques sont disponibles. C’est actuellement le langage le plus renommé dans le domaine du machine-learning et du datascience.

Cet outil est adapté pour des POC, parce que c’est principalement un outil de programmation de haut niveau, qui peut néanmoins sous-traiter des calculs très lourds à des bibliothèques écrites en C/C++ (langage de plus bas niveau).

Quelques livres de formation/ initiation :

* (ZIADE TAREK, 2009)
* (BEYELER MICHAEL, 2017)
* (LUTZ MARK, 2017)

# Etude sur des fichiers pdf avec texte disponible

A l’heure de cette première étude, nous ne disposions pas d’un lecteur Python pour le texte d’un pdf. En conséquence, nous avons pris des pdf sans texte qui avaient été OCRisés sur Cindoc, et donc le texte était disponible dans un fichier .txt séparé.

## Collecte de données (collecte de documents)

### Les documents dits ‘SciDoc’

Afin de simplifier le POC, nous allons réaliser un détour par la base de données de (e)Research appelée Kardex, qui est logée dans Cindoc. Elle ne contient *que* des documents scientifiques, tirés de journaux scientifiques, de congrès, ou de chapitres de livres. Cette collection fait l’objet d’un traitement OCR semi-automatisé depuis 2012 avec FineReader, si bien qu’il est possible de récupérer, pour ces documents, les fichiers .TXT qui contiennent les mots reconnus. Cet OCR n’est pas très performant, car il présuppose une langue anglaise, et délivre aussi parfois des caractères illisibles, issus d’une reconnaissance imparfaite. Mais nous verrons qu’il suffit largement pour ce POC.

Nous avons donc collecté **284 documents** .TXT, rentrés dans la base entre le 1er juillet 2016 et le 3 novembre 2016 correspondant à 284 pdf scientifiques, et nous les avons « taggués » comme scientifiques dans une variable fixée égale à « SciDoc »

### Les documents dits ‘NonSciDoc’

Les documents non-scientifiques (en format .TXT) ont été tirés d’anciennes archives de maintenance, telles qu’on peut affirmer sans erreur que ce ne sont pas des documents scientifiques : **233 documents**. L’idéal aurait consisté à prendre non pas des fichiers .TXT tirés d’un contexte si différent, mais de les tirer du réseau directement, et de les classifier manuellement. Malheureusement le format .TXT n’est plus utilisé aujourd’hui, et la lecture sous Python des documents DOC ou DOCX ne m’est pas encore connue.

Ces documents, à coup sûr non scientifiques, sont taggués « NonSciDoc »

## Calcul des traits caractéristiques (features) sur base .TXT

En raison de la petitesse de la base de connaissance liée à la lecture de fichier .txt uniquement, nous ne pouvons travailler en « deep learning ». Il nous faut donc guider le programme Python qui analyse les fichiers, en lui donnant les caractéristiques textuelles qui indiquent la *scientificité* vraisemblable des documents. Cette phase s’appelle le *Training*. Ces caractéristiques se déterminent assez intuitivement par lecture manuelle de la première page des documents pdf, et en lisant le fichier .TXT que Cindoc a associé à ces fichiers.

Il y a plusieurs indices de *scientificité :*  les indices « assez certains », et les indices « incertains ». Le classificateur de Python va calculer la capacité séparative de ces différents indices.

On citera, pour exemple :

* la présence de mots-indices d’un document publié : *published-by, approved by, copyright* *department of., «  toute reproduction sans »* etc..
* la présence de mots indices d’un document scientifique ou d’une norme : *abstract, researchgate, elsevier, « indice de classement », « normes en lignes »*
* la succession des deux mots « VOL » et « NO » qui ressemble à une collation bibliographique (ex: ***vol.*** *25,* ***no*** *6, pp.125-156)*
* la présence d’un nom de revue connue et différenciant : ‘*ISIJ International, « CLEFS CEA », mais pas « hydrometallurgy »*

On s’aperçoit par la pratique, qu’à mesure de la programmation/vérification des critères sur l’ensemble des documents scientifiques, le nombre des documents scientifiques ne satisfaisant aucun des critères diminue drastiquement. A la fin, il ne reste d’ailleurs que des documents dont le caractère scientifique est plus flou à formaliser, mais qu’on reconnaîtrait sans doute par analyse d’image (chapitres de livres, par exemple).

Même si cela n’a pas été fait, on peut aussi, dans les documents non-scientifiques, trouver des caractéristiques rendant peu vraisemblable la *scientificité*[[6]](#footnote-6) d’un document (ex : *curriculum, technical offer etc*). Mais pour explorer cette voie, il faudra réentraîner le classifieur sur une base documentaire réelle.

En résumé, un vecteur de caractéristiques contient 5 caractéristiques booléennes, qu’on a dénommées *science\_hint, contains\_edited, contains\_name\_journal, is\_a\_norm, contains\_RDCenter*, qui seront utilisées pour la classification.

## Developpment et evaluation de Modèles de « machine learning »

### Evaluation avec l’algorithme *naïve bayes*

Les deux bases de données précédentes « *SciDoc* » et « *NonSciDoc*» sont fusionnées, mélangées aléatoirement. Un programme va sélectionner 80% de cette base pour jouer le rôle « training ». Une fois le classificateur optimisé par l’algorithme Naïve Bayes portant sur ces 80%, on l’utilise sur les 20% restants de la base pour tester, et l’on voit si les résultats de classification automatique recoupent – *ou pas* – la classification manuelle prise en référence.

* En Python, la formation au module Naive Bayes se trouve en bibliographie : <http://www.nltk.org/book/ch06.html> (Python)

La performance du modèle est mesurée, pour l’instant, par le taux de bonne classification dans les deux catégories possibles : *SciDoc* et *NonScidoc*. Ainsi, une performance de 1 est parfaite, tandis qu’une performance de 0.95 atteste d’une erreur de classification de 5%, aussi bien par faux positifs que faux négatifs.

Le processus de mélange des 517 fichiers, de façon à créer un ensemble de référence (trainingset) et un ensemble test (testingset) est renouvelé de nombreuses fois, ce qui permet de tester toute une série de classificateurs, de façon à présenter la performance intrinsèque de la méthode comme une variable aléatoire. On trouve, par un usage répété sur 1000 mélanges successifs sous Python, des valeurs comprises entre 0.94 et 0.98 (1 étant la perfection)

Mélange (shuffle) de la base qui fait 517 fichiers

Nous plaçons la coupure au document numéro 414

Ce qui donne : 414 documents pour le Trainingset, et 103 documents pour le Testingset

Voici le résultat du test sur les 103 derniers

**0.9611650485436893**

Voici les critères les plus pertinents

Most Informative Features

science\_hint = 'yes' SciDoc : NonSci = 31.0 : 1.0

contains\_edited = 'no' NonSci : SciDoc = 3.9 : 1.0

science\_hint = 'no' NonSci : SciDoc = 3.6 : 1.0

contains\_name\_journal = 'no' NonSci : SciDoc = 1.2 : 1.0

is\_a\_norm = 'no' NonSci : SciDoc = 1.1 : 1.0

contains\_RDCenter = 'no' NonSci : SciDoc = 1.1 : 1.0

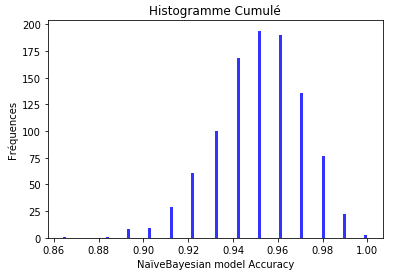
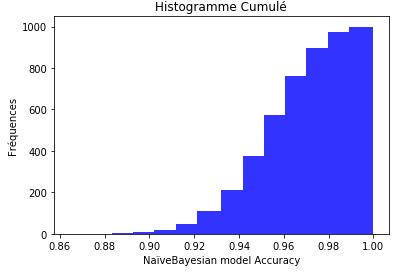
 

Figure 2. L’histogramme et l’histogramme cumulé du taux de bonne classification répété en 1000 fois

On peut donc affirmer que la méthode ainsi calibrée à partir de d’une base de 517 documents taggués manuellement en « *SciDoc* » pour les uns, et « *NonSciDoc* » pour les autres, admet une reproducibilité par machine learning de 95% sans erreur [de 90% à 99% selon l’ensemble de vérification]

### Evaluation avec des modèles d’apprentissage automatique avancés

Dans cette partie, nous utilisons des algorithmes d’apprentissage automatiques plus avancées, y compris « *Support Vector Machine* », « *Random Forest Classification* » et « *Gradient Boosting Classification* » pour évaluer la précision de classification des documents. Les hyper-paramètres des algorithmes ont été ajustés pour trouver l’ensemble optimisé à l’aide de méthodes de « extensive grid search » avec « cross validation ».

Le Tableau 1 résume les résultats obtenus par ces algorithmes. Le tableau montre que « *Random Forest Classification* » et « *Gradient Boosting Classification* » ont fourni une précision légèrement supérieure à celle des modèles de Naïve Bayes avec une moyenne de 0.951 contre 0.950. Cela signifie que les caractéristiques extraites sont assez discriminantes pour les deux types de documents : SciDoc et NonSciDoc.

|  |  |
| --- | --- |
| **Machine learning models** | **Testing Accuracy (mean ± std)** |
| Naïve Bayes (nltk) | 0.950 ± 0.012 |
| Gaussian Naive Bayes | 0.903 ± 0.019 |
| Multinomial Naive Bayes | 0.945 ± 0.016 |
| Support Vector Machines | 0.947 ± 0.013 |
| Random Forest Classification | 0.951 ± 0.012 |
| Gradient Boosting Classification | 0.951 ± 0.012 |

Tableau 1. Taux de bonne classification pour le « testingset » obtenus par différents algorithmes d’apprentissage automatique pour les documents .txt , répété en 100 fois

# Etude COMPLEMENTAIRE sur des pdf natifs

## Collecte de données (collecte de documents)

Un programme Python a été développé pour sélectionner tous les fichiers pdf de T/Project/Collaboratifs. Ensuite, nous les avons évalués manuellement et classés en deux groupes de documents scientifiques (*SciDoc*) et non scientifiques (*NonSciDoc*). En conséquence, 109 et 64 documents PDF natifs ont été étiquetés comme des documents scientifiques et non scientifiques, respectivement.

## Extraction de caractéristiques

### Lire le contenu du fichier pdf

Avant d'extraire les caractéristiques, nous devons développer un programme pour lire le contenu des documents pdf et extraire le contenu des premières pages qui sera utilisé pour l'extraction des caractéristiques. Le programme affichera également le nombre de pages du document, ce qui est considéré comme une caractéristique pour la classification.

Un programme est écrit sur la base de la bibliothèque Python pdfminer qui se concentre sur l'obtention et l'analyse des données de textes.

### Extraction de carateristiques

Après avoir obtenu le contenu de la première page du document, tous les caractères de ponctuation ont été supprimés. La liste des caractères de ponctuation est la suivante :

ponctuations = ['(', ')', ';', ':', '[', ']', ',', '.', '“', '”']

Nous avons décidé d'extraire des caractéristiques plus générales en comptant le nombre d'occurrences des mots suivants :

features = ['abstract', 'al.', 'article', 'authors', 'by', 'doi', 'index', 'institut', 'institute', 'introduction', 'journal', 'keywords', 'laboratory', 'manuscript', 'paper', 'par', 'publication', 'published', 'rapporteur', 'received', 'rev.', 'university']

Ces mots ont été soigneusement choisis en examinant les documents scientifiques. En fait, nous commençons par une liste de plus de mots, puis nous calculons la corrélation entre le nombre d'occurrences de ces mots avec les documents cibles (i.e., *SciDoc* ou *NonSciDoc*) afin de sélectionner la liste de 22 mots qui ont une valeur de corrélation élevée pertinente. Enfin, un jeu de caractéristiques contient 23 (22 mots plus le nombre de pages du document) caractéristiques représentant chaque document.

## Developpment et evaluation de Modèles de « machine learning »

Nous faisons les tests avec le même principe : 80% de fichiers servant à entraîner, et 20% de fichiers servant à tester le modèle, tel que décrit dans la sous-section 2.3.1. Tableau 2 résume les résultats.

Le tableau montre que les algorithmes d’apprentissage automatique avancés, notamment « Random Forest Classification », surpassent les algorithmes naïve Bayes. C’est parce que le nombre de caractéristiques extraites est plus grand (i. e. 23) que le nombre de caractéristiques extraits pour les documents txt (i. e. 5). En plus, plusieurs documents non scientifiques tels que les compte-rendus de réunions, les présentations, les rapports, etc. pourraient avoir les mêmes caractéristiques que les documents scientifiques.

|  |  |
| --- | --- |
| **Machine learning models** | **Testing Accuracy (mean ± std)** |
| Gaussian Naive Bayes | 0.872 ± 0.047 |
| Multinomial Naive Bayes | 0.715 ± 0.068 |
| Support Vector Machines | 0.908 ± 0.045 |
| Random Forest Classification | 0.912 ± 0.047 |
| Gradient Boosting Classification | 0.903 ± 0.047 |

Tableau 2. Taux de bonne classification obtenus par différents algorithmes d’apprentissage automatique pour les documents pdf natif, répété en 100 fois.

Les résultats indiqués dans les tableaux 1 et 2 indiquent que « Random Forest Classification » est le meilleur algorithme d’apprentissage automatique pour la classification des documents. La précision sera améliorée en augmentant le nombre de documents (SciDoc et NonSciDoc), actuellement 173. Ce sera une perspective de poursuite de l’étude.

# Perspectives d’utilisation

Nous voyons donc qu’il est relativement facile de développer différentes approches pour estimer, pour chaque fichier du réseau, une probabilité d’appartenance à la catégorie ‘scientifique’ sur la base de calculs plus ou moins évolués : soit sur des informations ‘texte’, et éventuellement sur des informations graphiques ou ’images’.

En termes de processus informatique, on peut alors schématiser de la façon comme Figure 3.

Pdf

à texte protégé

Pdf

scan pur

Pdf

natif

Pdf scan pur, OCR déjà fait

Extraction PNG première page

OCR

Extraction du text ou traitement d’image

Machine learning algorithms

Indexation (e) Research

Base de donnée cherchable

Identifier : titre, auteurs, année, source, mot-clés

Feature extraction

Figure 3. Processus de traitement

## Documents du réseau Eramet Research

Potentiellement, le 25/06/2018, le répertoire T:\PROJETS, qui a moins de deux ans d’existence, contient 54000 fichiers pour 157 Go.

A l’aide d’un logiciel sous Python capable de lire les pdf, de reconnaître le texte inclus ou bien de réaliser un OCR à la volée, il sera donc relativement facile de repérer ces documents avec un taux d’erreur raisonnable. Qui plus, est, l’analyse manuelle des erreurs de classification, réalisée par échantillonnage, permettra d’améliorer encore le processus de reconnaissance.

## Documents du réseau Eramet Ingénierie

Eramet Ingénierie dispose de deux répertoires géants contenant des documents techniques triés selon l’arborescence.

ST:\ : 104.000 fichiers, pour 200 Go.

W:\ : plusieurs millions de fichiers, 6,6 To (= 6600 Go)

Ces répertoires sont devenus si importants depuis tant d’années, qu’aujourd’hui, en gros, seules les personnes ayant elles-mêmes rentré les fichiers dans ces répertoires sont capables de les retrouver, ce qui fait que le partage documentaire est plus *théorique* que *pratique*.

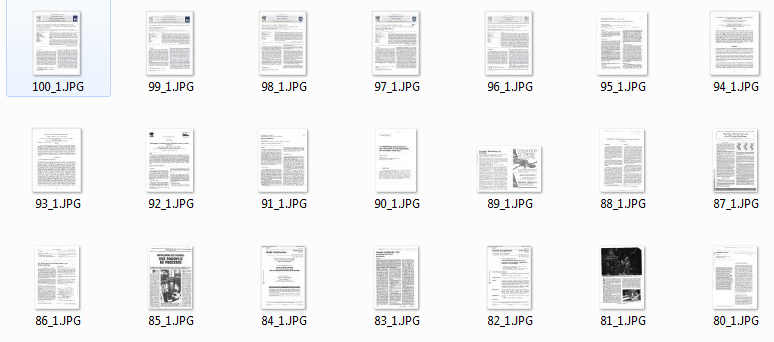
Les Techniques de l’Ingénieur font état de ce potentiel : (ESPINASSE BERNARD, BELLOT PATRICE)

## Quelques risques néanmoins repérés

La classification manuelle de documents pris au hasard sur le réseau T:\Projet fait apparaître un risque à ne pas négliger : des documents personnels, liés aux voyages des collaborateurs, s’y trouvent aussi (photocopies de passeports, remboursements de frais, achats de billets). Il s’agira, en même temps que la classification scientifique, de classifier aussi selon un axe « personnel » ce genre de documents personnels, de façon à ne pas prendre le risque d’intégrer dans la base scientifique des documents personnels qui seraient passés à travers les mailles du premier filet de qualification automatique de scientificité.

## Calcul des traits caractéristiques a partir d’images

Voici, pour exemple, à quoi ressemblent les images des documents pdf analysés jusqu’à présent par le texte.



Des critères visuels ou graphiques, signes de la scientificité d’un document pourraient être également être utilisés

* la présence d’un logo en haut à droite. On pourrait même reconnaître des logos précis.
* la présence de deux colonnes justifiées ou plus, au moins sur une certaine partie du document.
* un extrait de livre se détecte par l’alternance du côté des numéros de page, ainsi qu’un niveau de gris moyen en général supérieur)
* la présence d’un texte de plusieurs lignes en italique, trahissant l’abstract
* une liste des auteurs se reconnaît à la présence de nombreux points et de lettres esseulées sur une ligne unique (prénom/1°ere lettre + nom ex : K. MORX, SAREL G.)

Cet développement possible de la classification n’a pas été abordé pour l’instant. On peut raisonnablement penser que le taux d’erreur de classification (= 1-capacité discriminante) passerait de 5% à 2% ou 3%.

En plus de ces méthodes, la possibilité d’une méthode en *deep learning* sur des images est attestée : (HARLEY ADAM W., UFKES ALEX, DERPANIS KONSTANTINOS G., 2015), mais nous devons prendre le temps de bien évaluer la quantité d’images nécessaires pour nourrir une telle méthode.

# Prochaines étapes. Pré-ConcLusion

On l’a vu, la simple analyse de texte, éventuellement complétée d’une analyse des images, soit guidée par des « features » caractéristiques, soit par le deep learning, est capable de réaliser une collecte efficace des documents scientifiques. (RASCHKA SEBASTIAN, MIRJALILI VAHID, 2017). La présente étude n’est qu’un POC, basé sur des méthodes statistiques simples. L’outil des réseaux de neurones est aussi disponible (ENGELBRECHT ANDRIES P., 2007).

Afin de rendre la faisabilité plus certaine, il convient de développer au programme Python testé ici d’autres facultés essentielles

* Augmenter le nombre documents étiquetés manuellement comme *SciDoc* et *NonSciDoc*
* La capacité, le cas échéant, de réaliser un OCR sur un pur pdf-image.
* La capacité d’extraire une ou deux pages en format bmp pour la confier à un analyseur d’images, pour en tirer des « features » supplémentaires et ainsi améliorer la séléctivité du programme.
* La capacité de se mettre en connexion base de donnée avec CINDOC afin d’importer la signature unique des fichiers (méthode SHON), afin d’enlever tous les doublons : en effet nous l’avons dit, la moitié des bibliographies sont réalisées sur la base interne (e)Research.
* Et, étape ultime, d’intégrer ce processus dans un programme Customermatrix (=polyspot) d’indexation du réseau pour fournir des résultats filtrés selon le critère opérationnel « SciDoc », voire aussi tout autre paramètre « TechDoc », pour les ingénieurs d’Eramet Ingénérie. Cette étape serait, bien sûr, encore améliorée – mais c’est un sujet en soi – si les titres, auteurs, année, source et mot-clés peuvent être reconnus et lus dans chaque fichier.

# Autres applications possibles

Une telle classification peut aider toute Direction ou toute organisation d’Eramet à ranger ses affaires « sans les ranger ».

* Reconnaissance des factures, des comptes rendus de réunions, des spécifications d’achat,
* Des contrats divers, etc.

Python pourrait aussi permettre d’effectuer une analyse d’ensemble sur les fichiers d’un projet, contrôler le déploiement d’un projet selon la méthode attendue, etc...

# Bibliographie

**ESPINASSE BERNARD, BELLOT PATRICE** Introduction au Big Data. Opportunités, stockage et analyse des megadonnées [Revue] // KARDEX. - pp. H6040 P.1-11, LES TECHNIQUES DE L'INGENIEUR, (E)R. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=KARDEX&statement=NO=26204&view=view2.

**INTERNATIONAL ATOMIC ENERGY AGENCY** Control and Management of Radioactive Material Inadvertently incorporeated into Scrap Metal. Proceedings of an International Conference Tarragona, Spain, 23-27 February 2009 [Revue] // BIB. - 2009. - p. (E)R .. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=BIB&statement=NO=10034&view=view2.

**KIM SEON-HYO, FRUEHAN R.J.** Physical Modeling of Gas/Liquid Mass Transfer in a Gas StirredLladle [Revue] // KARDEX. - 1987. - pp. METALLURGICAL TRANSACTIONS B, VOL. 18B P.673-680, A.3.1.2. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=KARDEX&statement=NO=06607&view=view2.

**KLUTH T., QUADE U., LEDEBRINK F. W.** Recycling of Uranium and Plutonium Contaminated Metals From decommissioning of the Hanau Fuel Fabrication Plant [Revue] // KARDEX. - 2003. - pp. P., WM' 03 CONFERENCE FEB 23-27, 2003, TUCSON,AZ, (E)R. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=KARDEX&statement=NO=25237&view=view2.

**LENGAGNE LAETITIA** Réunions du 14 & 16 Novenmbre 2016 Echange sur les Techniques de Mesures Nucléaires avec NUVIA (n° d'Affaire Eramet Ingénierie : Projet 2530.B) [Revue] // RAPI. - 2016. - pp. 52.16.056\_CRR\_V01-LL ., 1.G.PYR0001-03, 139/16 ., 2530.B .. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=RAPI&statement=NO=17306&view=view2.

**Python** Chap 6 Learining to Classify Text [Ouvrage]. - http://www.nltk.org/book/ch06.html.

**SCHLIENGER ERIC M., BUCKENTIN JOANNA M., DAMKROGER BRIAN K.** Melt Processing of Radioactive Waste. A technical overview. [Revue] // KARDEX. - 1997. - pp. P., SANDIA, (E)R. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=KARDEX&statement=NO=25236&view=view2.

**ZIEMER HEINZ** Contribution à l'Etude des Méthodes de Calcul des Fours d'Electrométallurgie [Revue] // KARDEX. - 1962. - pp. JOURNAL DU FOUR ELECTRIQUE ET DES INDUSTRIES ELECTROCIMIQUES, 8 P.271-282, N, H.6.3. - http://cindoc.era:8080/cindocwebjsp/default.jsp?bc=simplequery&archive=KARDEX&statement=NO=06619&view=view2.

# Annexe I

## Calcul de la liste des fichiers a lire

#%% Fonction pour Déterminer une liste des fichiers à lire. La sélection est faite sur "txt"

# qui est là juste pour limite le nombre des fichiers.

def Détermine\_liste\_fichiers(chemin\_de\_base,tag):

liste\_fichiers=[]

for root,dirs,files in os.walk(chemin\_de\_base):

for f in files:

if f[len(f)-4:len(f)]=='.txt' and ('' in f):

liste\_fichiers.append((root,f,tag))

print(len(liste\_fichiers),' fich. dans ',chemin\_de\_base)

return(liste\_fichiers)

## Lire le contenu du fichier pdf

def ReadPDFFile(fname, pages=None):

'''Read pdf file and return the text of specified page and number of pages

'''

if not pages:

pagenums = set()

else:

pagenums = set(pages)

output = io.StringIO()

codec = 'utf-8'

imagewriter = None

manager = PDFResourceManager()

laparams=LAParams()

laparams.all\_texts = True # Correspond to -A option

device = TextConverter(manager, output, codec=codec, laparams=laparams,

imagewriter=imagewriter)

#device = TextConverter(manager, output, laparams=laparams)

interpreter = PDFPageInterpreter(manager, device)

infile = open(fname, 'rb')

# Read the number of pages

parser = PDFParser(infile)

document = PDFDocument(parser)

nPages = resolve1(document.catalog['Pages'])['Count']

# Inteprete the text of the document

for page in PDFPage.get\_pages(infile, pagenums):

interpreter.process\_page(page)

infile.close()

device.close()

text = output.getvalue()

output.close

return text, nPages

## Extraction des features pour document txt

#%% recherche des features qui signent un doc scientifique

def science\_features(liste\_de\_mots):

resul={'science\_hint' : 'no', 'contains\_edited' : 'no', 'contains\_name\_journal' : 'no', 'is\_a\_norm' : 'no', 'contains\_RDCenter' : 'no'}

liste\_de\_mots\_dc=[mot.lower() for mot in liste\_de\_mots]

# voir si le mot volume apparaît

if 'volume' in liste\_de\_mots\_dc:resul['science\_hint']='yes'

# à faire : si dans la première page on trouve VOL ..NO ...annee (il faut éviter les références de la biblio !)

#print(liste\_de\_mots\_dc, resul)

# analyse des mots-simples

if 'sciencedirect' in liste\_de\_mots\_dc:resul['contains\_edited']='yes'

if 'elsevier' in liste\_de\_mots\_dc:resul['contains\_edited']='yes'

if 'isbn' in liste\_de\_mots\_dc:resul['contains\_edited']='yes'

if 'issn' in liste\_de\_mots\_dc:resul['contains\_edited']='yes'

if 'proceedings' in liste\_de\_mots\_dc:resul['contains\_edited']='yes'

if 'researchgate' in liste\_de\_mots\_dc:resul['contains\_edited']='yes'

if 'abstract' in liste\_de\_mots\_dc:resul['science\_hint']='yes'

if 'received' in liste\_de\_mots\_dc[0:300] and 'accepted' in liste\_de\_mots\_dc[0:300]:resul['contains\_edited']='yes'

# analyse des successions de deux mots

for i in range(len(liste\_de\_mots\_dc)-1):

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='metallurgical transactions':resul['contains\_name\_journal']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='isij international':resul['contains\_name\_journal']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='published by':resul['contains\_edited']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='edited by':resul['contains\_edited']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='published online':resul['contains\_edited']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='report submitted':resul['contains\_edited']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='research report':resul['science\_hint']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='chief editors':resul['contains\_edited']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='department of':resul['contains\_edited']='yes'

# analyse des successions de 3 mots

for i in range(len(liste\_de\_mots\_dc)-2):

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='indice de classement':resul['is\_a\_norm']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='stahl und eisen':resul['contains\_name\_journal']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='normes en ligne':resul['is\_a\_norm']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='toute reproduction sans':resul['contains\_edited']='yes'

# analyse des successions de 4 mots

for i in range(len(liste\_de\_mots\_dc)-3):

if liste\_de\_mots\_dc[i]=='vol' and (liste\_de\_mots\_dc[i+2]=='no' or liste\_de\_mots\_dc[i+2]=='nr'):resul['science\_hint']='yes'

# ce dernier doit être amélioré pour capture plus flou et sur la première page seulement

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='stahl und eisen':resul['contains\_name\_journal']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='normes en ligne':resul['is\_a\_norm']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]+' '+liste\_de\_mots\_dc[i+2]=='toute reproduction sans':resul['contains\_edited']='yes'

# voir si contient SINTEF et Trondheim

if 'sintef' in liste\_de\_mots\_dc and 'trondheim' in liste\_de\_mots\_dc :resul['contains\_RDCenter']='yes'

for i in range(len(liste\_de\_mots\_dc)-1):

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='swerea mefos':resul['contains\_RDCenter']='yes'

if liste\_de\_mots\_dc[i]+' '+liste\_de\_mots\_dc[i+1]=='clefs cea':resul['contains\_RDCenter']='yes'

# à faire : si dans la première page on trouve VOL ..NO ...annee (il faut éviter les références de la biblio !)

return(resul)

## Extraction des features pour des documents pdf natif

def ExtractFeatures (extracted\_text, textKeyDictInit):

'''Extract features from text

'''

textKeyDict = textKeyDictInit.copy()

extracted\_text\_r = extracted\_text.replace('\n', ' ') # remove '\n' character in the middle of string

extracted\_text\_low = extracted\_text\_r.lower()

tokens = word\_tokenize(extracted\_text\_low)

scannedPDF = 0

# Check if tokens is just in characters, or the word is split in characters

if tokens: # tokens is not empty

punctuations = ['(', ')', ';', ':', '[', ']', ',', '.', '“', '”']

tokens = [word for word in tokens if not word in punctuations]

if (len(tokens) > 100):

sizeRN = int(len(tokens)/30)

randNum = np.random.randint(20, size=sizeRN)

else:

sizeRN = len(tokens)

randNum = range(sizeRN)

lenTokens = 0

for i in randNum:

lenTokens += len(tokens[i])

if(lenTokens == sizeRN): #tokens are just characters

print('Recompute the tokens ...')

extracted\_text\_r = extracted\_text.replace('\n', '') # remove '\n' character in the middle of string

extracted\_text\_low = extracted\_text\_r.lower()

tokens = word\_tokenize(extracted\_text\_low)

from collections import Counter

count = Counter(tokens)

print(count.most\_common(20))

for key in textKeyDictInit.keys():

for countKey, countValue in count.items():

if key == countKey:

textKeyDict[key] = countValue

print(countKey, countValue)

else:

scannedPDF = 1

print('Tokens is empty, scanned pdf is read')

print(textKeyDict)

return textKeyDict, scannedPDF

#%%

textKeys = ['keywords', 'abstract', 'volume+number', 'introduction', 'authors', 'publication', 'index','no.', 'by', 'rev.', 'journal', 'jnl', 'laboratory', 'university', 'univ.', \

'al.', 'manuscript','tutorial', 'institute', 'research', 'doi', 'published', 'received', 'article', 'conference', 'paper', 'mot-cles', 'resumé', 'par', 'institut', 'rapporteur']

values = [0]\*len(textKeys)

textKeyDictInit\_temp = dict(zip(textKeys, values))

from collections import OrderedDict

textKeyDictInit = OrderedDict(sorted(textKeyDictInit\_temp.items()))

# Extract scientific document features

pathText = ["D:\\Work\\TextAnalysis\\DocClassification\\5-Experiments\\RandomSelectedDocs\\NSD",

"D:\\Work\\TextAnalysis\\DocClassification\\5-Experiments\\RandomSelectedDocs\\SD"]

data = []

noScannedPdf = 0

for sci in range(2):

filePath\_PDF = collectFilePath(pathText[sci], '.pdf')

for i in range(len(filePath\_PDF)):

#for i in range(10,15):

try:

(extracted\_text, nPages) = ReadPDFFile(filePath\_PDF[i], pages=[0]) #0 or 6

print('===========Index = {}=================='.format(i))

print(filePath\_PDF[i])

except:

continue

textKeyDict, scannedPDF = ExtractFeatures (extracted\_text, textKeyDictInit)

if scannedPDF:

noScannedPdf += 1

else:

temp = list(textKeyDict.values())

temp.extend([nPages, sci]) #use list.append() to append a single value, and list.extend() to append multiple values.

data.append(temp)

print('No of pages: {}'.format(nPages))

data\_np = np.array(data)

print('{0} scanned pdf out of {1} pdf files'.format(noScannedPdf, len(filePath\_PDF)))

## « Grid search » et « cross validation » pour trouver des parametres optimisés

dataTXT = np.array(databaseset)

# Grid search for finding optimized parameters

def report(results, n\_top=3):

for i in range(1, n\_top + 1):

candidates = np.flatnonzero(results['rank\_test\_score'] == i)

for candidate in candidates:

print("Model with rank: {0}".format(i))

print("Mean validation score: {0:.3f} (std: {1:.3f})".format(

results['mean\_test\_score'][candidate],

results['std\_test\_score'][candidate]))

print("Parameters: {0}".format(results['params'][candidate]))

print("")

X = dataTXT[:,:-1] # features

y = dataTXT[:,-1] # labels, 1: sci, 0: nonsci

# SVM classifier

print("---------SVC classifier -----------------------")

svc\_model = SVC()

svc\_parameters = {'kernel': ('linear', 'rbf', 'poly'), 'C': [1, 10, 100]}

svc\_gs = GridSearchCV(svc\_model, svc\_parameters, cv=5)

svc\_gs.fit(X, y)

svc\_gs.best\_params\_

report(svc\_gs.cv\_results\_)

# Random forest

print("---------RF classifier -----------------------")

rf\_model = RandomForestClassifier()

rf\_parameters = [ {'n\_estimators': [10, 50, 100, 1000], 'max\_features': [1, 3, 5]},

{'bootstrap': [False], 'n\_estimators': [1, 50, 100, 1000], 'max\_features': [1, 3, 5]}]

rf\_gs = GridSearchCV(rf\_model, rf\_parameters, cv=5)

rf\_gs.fit(X,y)

rf\_gs.best\_params\_

report(rf\_gs.cv\_results\_)

# GradientBoostingClassifier

print("---------GBM classifier -----------------------")

gbc\_model = GradientBoostingClassifier()

gbc\_parameters = {'n\_estimators': [10, 100, 1000], 'max\_features': [1, 3, 5], 'learning\_rate': [0.1, 0.2, 0.5]}

gbc\_gs = GridSearchCV(gbc\_model, gbc\_parameters, cv=5)

gbc\_gs.fit(X, y)

gbc\_gs.best\_params\_

report(gbc\_gs.cv\_results\_)

## Evaluation des modeles d’apprentissage automatique

#%% Block d'analyse de la base DATABASESET

acc\_nbc, acc\_svc = [], []

acc\_gnb, acc\_mnb, acc\_bnb = [], [], []

acc\_rfc, acc\_gbc = [], []

for i in range(100):

print('Mélange (shuffle) de la base qui fait',len(databaseset),' fichiers')

random.shuffle(databaseset)

print('Nous plaçons la coupure au document numéro',round(len(databaseset)\*80/100))

trainingset=databaseset[0:round(len(databaseset)\*TR/100)]

testingset=databaseset[round(len(databaseset)\*TR/100):]

print('Ce qui donne :',len(trainingset),'documents pour le Trainingset, et ',len(testingset),' documents pour le Testset')

NBC\_model = nltk.NaiveBayesClassifier.train(trainingset)

GNB\_model = SklearnClassifier(GaussianNB(), sparse=False).train(trainingset)

MNB\_model = SklearnClassifier(MultinomialNB(), sparse=False).train(trainingset)

BNB\_model = SklearnClassifier(BernoulliNB(), sparse=False).train(trainingset)

SVC\_model = SklearnClassifier(SVC(C=1, kernel='linear'), sparse=False).train(trainingset)

RFC\_model = SklearnClassifier(RandomForestClassifier(n\_estimators=100, max\_features=1), sparse=False).train(trainingset)

GBC\_model = SklearnClassifier(GradientBoostingClassifier(n\_estimators=100, learning\_rate=0.1, max\_features=1), sparse=False).train(trainingset)

acc\_nbc.append(nltk.classify.accuracy(NBC\_model,testingset))

acc\_gnb.append(nltk.classify.accuracy(GNB\_model, testingset))

acc\_mnb.append(nltk.classify.accuracy(MNB\_model, testingset))

acc\_bnb.append(nltk.classify.accuracy(BNB\_model, testingset))

acc\_svc.append(nltk.classify.accuracy(SVC\_model,testingset))

acc\_rfc.append(nltk.classify.accuracy(RFC\_model,testingset))

acc\_gbc.append(nltk.classify.accuracy(GBC\_model,testingset))

print('NBC Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_nbc), np.std(acc\_nbc)))

print('Gaussian NB Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_gnb), np.std(acc\_gnb)))

print('Multinomial NB Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_mnb), np.std(acc\_mnb)))

print('Bernoulli NB Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_bnb), np.std(acc\_bnb)))

print('SVC Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_svc), np.std(acc\_svc)))

print('RFC Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_rfc), np.std(acc\_rfc)))

print('GBC Accurary: {0:.3f} (average) and {1:.3} (std)'.format(np.mean(acc\_gbc), np.std(acc\_gbc)))

1. GED = Gestion Electronique de Documents [↑](#footnote-ref-1)
2. POC = Proof Of Concept = logiciel permettant de démontrer la possible programmation d’un concept [↑](#footnote-ref-2)
3. OCR = Optical Character Recognition = opération de reconnaissance du texte dans un fichier image. [↑](#footnote-ref-3)
4. *machine learning*, dans la suite du document [↑](#footnote-ref-4)
5. *deep learning* [↑](#footnote-ref-5)
6. néologisme désignant le caractère scientifique d’un document [↑](#footnote-ref-6)