CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

Rapport de projet

SPÉCIALITÉ : INFORMATIQUE

PARCOURS : Big Data et Intelligence Artificielle

par

BARIBEAUD Arthur, MONTIEUX Etienne, REVERSAC Paul, VIRONDEAU Mathilde

---

Extraire des données d'un livret de famille

Décembre 2021

---

le cnam
école d'ingénieur·e·s

Table des matières

Introduction

Objectif du projet

Le projet qui nous a été confié est l'étude et l'extraction de données dans des documents non structurés. Plus spécifiquement, les documents de notre dataset sont des livrets de familles, *"un document officiel, délivré dans plusieurs pays et consistant en un recueil d'extraits d'actes d'état civil relatifs à une famille*" source Wikipédia.
Un livret de famille se compose ainsi de plusieurs pages dans lesquelles il est possible de trouver les différents actes civils d'une famille, on y retrouve par exemple dans les premières pages des informations concernant les parents (nom, prénom, date de naissance, etc) puis dans les pages suivantes des informations concernant les enfants. Ainsi chaque page est composé d'un ou deux actes de naissance d'enfant. Ce document est important puisqu'il fait fois du nombre d'enfants d'une famille devant l'état lors de démarches administratives tel que l'attribution des allocations familiales.

Problématiques à traiter

L'étude et la recherche manuelle d'informations dans des livrets de familles peut s'avérer être une tâche fastidieuse, mais pourtant essentielle pour éviter toute fraude dans la déclaration du nombre d'enfants à charge. Une solution envisagée est donc d'automatiser ce process avec une pipeline de traitement NLP.
L'objectif de cette pipeline est alors de transformer les documents d'entrées (des images aux formats .pdf) en des données analysables, puis pour chaque document tenter de compter le nombre d'enfants. En sortie de cette pipeline, on souhaite donc obtenir pour chaque livret de famille une estimation du nombre d'enfants comparativement à ce que la famille a annoncé afin d'éviter toute fraude

Plan de travail

Préparation des données

Le dataset dont nous disposons pour effectuer cette analyse est constitué de 18 livrets de familles stockés au format pdf. Chaque document possède un nombre de pages quelconque allant de 2 à 5 et avec un nombre d'enfants allant de 0 à 4. Afin de préparer les données pour des analyses, nous avons dû procéder à une étape de data preprocessing afin de les rendre exploitables.

Préparation des images

La première étape a tout d'abord été de transformer les pdf en objet python Image (lib PIL et pdf2image) puis d'extraire de ces images les données sous formes de chaînes de caractères à l'aide de la librairie pytesseract. Cependant, les images de notre dataset sont

toutes de qualité et de format différents, il a donc été nécessaire de nettoyer les images avant de les transformer en chaînes de caractères. Pour cela, plusieurs techniques existent tel que la mise en niveau de gris, la rotation, la suppression de bruits, etc. Néanmoins, et comme précisé précédemment, les images de notre dataset sont toutes différentes, prises avec des prises de vue, des qualités d'image et des luminosités différentes. Ainsi, les étapes de nettoyages mises en place sont particulièrement complexes puisque ne s'adaptant pas à toutes les images, en en améliorant certaines mais en en détériorant d'autres.

Pour palier à cette problématique, nous avons mis en place un algorithme permettant de boucler plusieurs fois sur un même livret de famille en modifiant les paramètres de nettoyage à chaque itération. Cela permet ainsi d'optimiser l'extraction du nombre d'enfants puisque la détérioration des images entraine une incapacité à détecter les enfants pour notre algorithme. Ainsi l'itération sur laquelle l'algorithme détecte le plus d'enfant est désignée comme celle ayant les paramètres d'images les mieux adaptés et est donc gardée comme valeur finale.

Enfin, pour chaque image convertie en chaînes de caractères, on procède ensuite à une étape de nettoyage des strings. On normalise dans un premier temps les chaines de caractères en minuscule avant de splitter chacune d'entre elles en listes de jetons. Chaque jeton est ainsi traité à l'aide de pyspellchecker afin de corriger les éventuelles erreurs : exemples → "naissznces" sera transformé en "naissances".
Un point important à noter est que nos images, bien qu'ayant été nettoyées, comportent encore de nombreux "bruits" qui sont par la suite transformés en chaines de caractères. Une première approche pour nous a été de retirer ces bruits, puisqu'à première vue, non porteurs de sens pour notre analyse.
Un fait important nous est cependant rapidement apparu : les bruits détectés étaient pour la plupart les écritures manuscrites qui n'avaient pas réussi à être transformées en chaînes de caractères intelligibles. Ainsi, en retirant ces "bruits", nous perdions l'information cruciale de savoir si un acte de naissance avait été complété ou non, et ainsi de connaitre le nombre d'enfants.
Notre méthodologie d'approche a, de ce fait, complétement changé en prenant le parti de faire de ces "bruits" nos indicateurs principaux. Ainsi les étapes de nettoyage des chaines de caractères se sont simplement vues être constituées de deux tâches : normalisation en minuscule et normalisation de la grammaire pour les chaînes tapuscrits.

Méthode utilisée

Première approche : Machine Learning

La première méthode envisagée a été l'utilisation d'algorithme de type machine learning afin de tenter de prédire, pour chaque livret de famille, le nombre d'enfants. Cette première approche expérimentale visait simplement à observer les résultats que ces types de modèles pouvaient obtenir tout en faisant l'hypothèse que ceux-ci échoueraient dû aux "bruits" de nos documents. Pour ce faire nous avons transformé chaque document en "bag

of words” d’unigrame et bigrame afin de les entraîner sur différents modèles (SVM, Naïves Baies, KNN). Les premiers résultats obtenus se sont avérés très mauvais (entre 40% et 60% de prédiction correcte). L’hypothèse émise à cet instant est que les mauvais résultats peuvent être expliqués par une trop grande diversité du vocabulaire de notre bow, constitué principalement de bruits ('f’, ‘run’, ‘qe’, ‘eme’, ‘t’, ’ete’, ‘cm’, etc).

Une idée, pour pallier ce problème, a été de normaliser nos bruits sous un format commun “XXX”, ainsi notre vocabulaire serait radicalement réduit à nos données tapuscrites et à la chaîne “XXX”. Les résultats obtenus n’ont cependant pas validé notre hypothèse puisqu'ils sont restés pratiquement inchangés (entre 40 et 60%)

Les résultats restant décevant, la décision a été prise de changer d’approche en laissant de côté la méthode de machine learning et en se concentrant purement sur des méthodes de NLP.

Deuxième méthode : NLP

La seconde méthode envisagée a été de tourner le bruit, qui nous dérangeait jusqu’à présent, comme notre principal outil de détection d’enfant.

En effet, comme expliqué dans la partie “Préparation d’image”, nous avons constaté que les données tapuscrites étaient bien mieux retranscrites que les données manuscrites. Ainsi nos documents prenaient régulièrement la forme suivante : “*acte de naissance n° dzerz*”. On identifie ainsi ici en rouge les données tapuscrites et en bleu les données manuscrites. Ainsi nous sommes partis du principe qu’aucune donnée manuscrite n’est inscrite à la suite de données tapuscrites si aucun enfant n’est déclaré.

À la suite de ce constat, nous avons mis en place un ensemble de règles de décision afin de tenter de prédire pour chaque document si oui ou non un enfant était déclaré en partant de la règle suivante : si le document possède une chaine de caractère similaire à “acte de naissance n°” suivi d’une chaîne de caractères identifiée comme du bruit, alors un enfant est déclaré. Cette règle, bien qu’à première vue très simple, s’est avérée très efficace. Il a cependant été nécessaire d’effectuer quelques ajustements afin de l’optimiser. Nous avons par exemple constitué un dictionnaire des différentes formes que pouvait prendre la chaîne n° puisque le caractère “°” n’était pas toujours correctement retranscrit : [“n°”, “ne”, n*”, “n”]. Autre point important ayant permis la mise en place de cette méthode est le fait que l'outil d’ocr pytesseract sépare les deux pages d’un livret par un caractère d’échappement “\n”.

En effet, la méthode d’ocr de pytesseract analyse et transforme les images ligne par ligne, ainsi une ligne type ressemble à ceci : “Extrait de l’acte de naissance n° Extrait de l’acte de naissance n°” comme sur l’image ci-dessous :

Avec ce type de résultat, notre hypothèse de regarder le bruit à la suite d'une chaine "acte de naissance n°" n'aurait pas marché puisqu'à la suite, on y retrouve "Extrait" et donc notre algorithme aurait analysé ce résultat comme la présence d'un enfant. Cependant, la sortie de pytesseract s'est toujours avérée être de la forme "Extrait de l'acte de naissance n° \n Extrait de l'acte de naissance n°" pour chaque ligne. Cela nous a ainsi permis de spliter nos lignes selon le caractère d'échappement "\n" et ainsi permettre le bon fonctionnement de notre modèle de détection.
Une seconde règle de décision utile a également été d'analyser uniquement les premières lignes de chaque document puisque c'est dans ces dernières que l'on retrouve les informations utiles.

```
try:
    # Case : "naissance + CASE_NUMBER + noise
    if word == "naissance" and words[words.index(word) + 1] in CASE_NUMBER and len(
            words[words.index(word) + 2]) > 1:
        count_child += 1
    elif word == "naissance" and words[words.index(word) + 1] not in CASE_NUMBER and len(
            words[words.index(word) + 1]) > 1:
        count_child += 1
except:
    pass
```

Règle de décision principale

Résumé de la méthode retenue

On fournit en entrée un livret de famille au format PDF à notre algorithme. Puis, il effectue une première étape de nettoyage d'image. Une fois nettoyées, les images sont transformées en chaînes de caractères puis, de nouveau nettoyées.
Ainsi chaque livret de famille est décomposé en plusieurs images et chaque image est décomposé en une liste de string. Chaque string de la liste représente une ligne de l'image.

Pour chaque image d'un livret de famille, on utilise ensuite notre méthode de détection du nombre d'enfants. Ainsi pour chaque string de notre liste de ligne, on tente de détecter si une écriture tapuscrite de la forme "acte de naissance n°" est suivie d'une écriture manuscrite.

Pour chaque image traitée, dans le cas où le résultat obtenu est de 0 enfants, on émet l'hypothèse que l'étape de nettoyage d'image n'a pas fonctionné correctement. Dès lors, on réitère l'opération précédente sur de nouveaux paramètres de nettoyage. Si au bout de 3 itérations sur des paramètres de nettoyage différents, le résultat reste de 0 enfant, alors on s'arrête et on déclare que l'image ne contient aucun enfant.

Enfin, une fois l'ensemble des images d'un livret de famille traité, il est possible de donner le nombre d'enfants détecté.

Résultats obtenus

Pour vérifier la viabilité de notre modèle, nous avons fait tourner notre algorithme sur l'ensemble des livrets de famille en vérifiant manuellement les résultats obtenus avec la réalité. Cette tâche manuelle nous est permise du fait de la taille très restreinte de notre modèle.

Les résultats obtenus sont mitigés puisque seulement 62.5% des prédictions s'avèrent correctes. Cependant, nous pouvons valider la viabilité de notre modèle puisque les erreurs observées sur notre modèle sont principalement dû à des images de très mauvaises qualités que même les différents paramètres de nettoyage n'ont pas su améliorer. De ce fait, il s'agit de documents tout simplement inexploitables pour notre algorithme puisque même les données tapuscrites sont retranscrites comme du bruit.

Exemple d'un document inexploitable malgré différents nettoyages

En revanche pour les documents ayant été correctement nettoyés et donnant donc des résultats tapuscrits corrects, notre algorithme fonctionne très bien.

Conclusion

Le projet d'analyse des livrets de famille s'est avéré beaucoup plus compliqué que prévu, et ce, pour une raison différente de que ce que nous imaginions et de ce qu'il nous a déjà été donné de rencontrer dans d'autres projets. En effet, nous nous attendions à devoir essentiellement jouer sur nos paramètres d'algorithmes de décision ou encore sur les paramètres de nos modèles de machine learning. La réalité a été, que nous avons dû concentrer nos efforts sur le nettoyage de nos données qui étaient de très faibles qualités. Nous avons pour cela dû revoir nos ambitions en changeant notre méthodologie : passer du ML à la NLP ou encore en modifiant les paramètres de nettoyage pour tenter de s'adapter à chaque image.
Bien que compliqué et donnant des résultats mitigés (n'étant pas de notre ressort au vu de la qualité des données) ce projet a été très enrichissant en nous faisant comprendre l'importance et la complexité que représentent les problématiques d'OCR et les étapes de préprocessing sur les données du monde réelles.

Les perspectives d'améliorations nous paraissent ainsi principalement axées sur le nettoyage des images et sur les techniques d'OCR en utilisant des outils propriétaires et différents de Tesseract.