Création d’un dataset pour finetuné llama-3b

# Préliminaire) Récupérer les scans

Pour récupérer les scans, j’utilise l’outil EDM Extraction Tool, situé à [\\EU50APVD350\ClickOnces\EdmExtractionTool](file:///\\EU50APVD350\ClickOnces\EdmExtractionTool) ou [\\Eu51fiv02\ITR02\_DE\_elims\_NG\EDM Extraction Tool DMA v2](file:///\\Eu51fiv02\ITR02_DE_elims_NG\EDM%20Extraction%20Tool%20DMA%20v2) sur Citrix et créé par Ramees Machingal ([ramees.machingal@it.eurofinseu.com](mailto:ramees.machingal@it.eurofinseu.com)) . Il permet d’extraire les scans sur EUFR110 et EUFR93.

# I)Récupération des codes samples des scans

Script à lancer: process\_datas\_for\_dataset\_creation.py

**Ce script va filtrer les différents fichiers extraits via l’EDM et les déplacer dans un autre fichier. De plus, un fichier csv sera généré contenant les codes échantillons (sampleCodes) et les chemins associées (path). D’autres informations optionnelles sont également générées pour garder un meilleur suivi des informations.**

Exemple :

|  |
| --- |
| python /content/process\_datas\_for\_dataset\_creation.py "/content/drive/MyDrive/Data/data\_test/test\_original" "/content/output" |

Le **premier argument** est le dossier contenant les scans.

Le **deuxième argument** est le dossier vers lequel seront transférés les scans ainsi que le csv appelé processed\_pdfs.csv .

Les autres arguments sont : --date\_extraction, --extraction\_window\_start, --extraction\_window\_end, --lab et –env.

# II)Faire une demande à l’IT pour extraire les estimates

Demander l’extraction du code SQL suivant pour les codes échantillons extraits précédemment :

|  |
| --- |
| USE **<NG>** select distinct s.samplecode, o.OrderCode, o.OrderType, f.OriginalFractionCode , tpr.PackageCode , tpr.TestCode, t.TestName , rr.ParameterCode, prm.ParameterName , rr.DisplayValue, rr.ResultUnit --, CASE WHEN isnull(ev.Value,'') <> '' THEN 'yes' ELSE 'no' END HasNutritionalClaim , CASE WHEN isnull(nc.Value,'') = 'YES' THEN 'yes' ELSE 'no' END HasNutritionalClaim , ev.Value as EstimatedValue, ev.NumeratorUnit as EstimatedNumeratorUnit, ev.DenominatorUnit as EstimatedDenominatorUnit from BIODS\_sample s inner join BIODS\_Order o on o.orderid=s.orderid inner join BIODS\_Fraction f on f.SampleId=s.SampleId inner join BIODS\_TestPRepetition tpr on tpr.FractionId=f.FractionId inner join BIODS\_ReportableResult rr on rr.TestPRepetitionId=tpr.TestPRepetitionId and rr.Version=1 --published left join BIODS\_ExpectedValue ev on ev.FractionId=f.FractionId and rr.ParameterCode=ev.ParameterCode left join BIODS\_ExpectedValue nc -- with nutritional claims ? on nc.FractionId=f.FractionId and nc.ParameterCode='HE0001EA' inner join NG\_ITAAG010\_PRD42\_CONFIG\_ODS.comlims.test t on t.testcode=tpr.TestCode inner join NG\_ITAAG010\_PRD42\_CONFIG\_ODS.e5.Parameter prm on prm.ParameterCode=rr.parameterCode where s.SampleCode in ( **<LIST SAMPLE CODE>**) |

Il faut remplacer NG par la table où sont stocker les informations : NG\_EUFR110\_PRD42\_BI\_ODS ou NG\_EUFR93\_PRD42\_BI\_ODS.

# III)Avec les estimates, créer une nouvelle table

La table créée par l’IT est de la forme : Sample code – Package code nom paramètre – estimate. On va la transformer en la table : Sample code – Energie (kJ) – Energie (kCal) – etc.

De plus, on a en profiter pour trier la table et de sélectionner que les codes samples où toutes les valeurs de la table on était renseignées. Pour ce faire, on utilise les codes packages.

Exemple :

|  |
| --- |
| python extract\_NV\_from\_excel.py "/content/drive/MyDrive/Data/set4-pdf/extraction-set-4.xlsx" "/content/drive/MyDrive/Data/set4-pdf/datas.csv" 0 --output "/content/here.csv" |

Le **premier argument** est l’Excel issu de l’extraction.

Le **deuxième argument** est le csv généré dans le I) avec les colonnes ‘**path’** et ‘**sampleCodes’.**

**Le troisième argument** **est le chemin où sera généré le dossier de sortie.**

**Attention, cette méthode est la meilleure que j’ai pu faire, mais n’est pas parfaite. Il y a 3 problèmes majeurs :**

* **Les erreurs de codages (qui ne sont pas totalement négligeable, il y a une erreur tout les 10/15 scans)**
* **Les fibres ne sont pas toujours indiquées (la table sur le scan peut contenir la paramètre fibre, mais il n’est parfois pas demandé par le client, donc l’estimate n’est pas enregistré sur la base de données)**
* **Enfin, il y a des scans, où la table nutritionnelle n’est pas présente ou pas lisible.**

**Ainsi, je recommande de checker les scans à la main. Mais cette tâche est extrêmement longue et pénible.**

# IV)Enfin, créer la dataset pour finetuné llama 3-8b

Le dataset créé est sous la classe DatasetDict de Datasets, avec les colonnes : ‘prompt’, ‘input’, ‘output’, ‘path’ et ‘instruction’. Pour précision, ‘prompt’ comprend ‘instruction’, ‘input’ et ’output’. Pour le finetuning, seulement la colonne ‘prompt’ est utilisée.

Exemple :

|  |
| --- |
| python dataset\_llama3\_creation.py " input.csv" "file/output/or/hugging\_face/"  --id\_starting\_estimate\_column 3  --lenght\_nutrition\_table 40  --save\_to\_disk  --tokens\_for\_transformer “” |

Le **premier argument** est la table créée dans la partie III.

Le **deuxième argument** est le path où sera enregistré le dataset.

**Enfin les arguments optionnels sont :**

* *id\_starting\_estimate\_column*: l’id de la colonne à partir de laquelle commence les paramètres nutritionnelles, normalement 3.
* *lenght\_nutrition\_table*: le nombre de paramètres
* *save\_to\_disk* : si la sauvegarde est locale
* *push\_to\_hub*: ou si la sauvegarde est en ligne sur HuggingFace
* *tokens\_for\_transformer*: si la sauvegarde est sur HuggingFace, il faudra renseigner ses tokens pour l’identification.

# V)Possibilité d’améliorer le dataset via le cropping

Une fois le dataset créé, celui-ci peut-être améliorer via l’algorithme de Table Détection. En effet, pour chaque scan, si une table ou plusieurs sont détectées, alors une ligne sera ajoutée comprenant le texte lu par l’ocr sur la ou les différentes tables (input + prompt) et le nouveau chemin de l’image, les autres informations seront inchangées (output et instruction). En effet, il est préférable d’entraîner le modèle sur l’ensemble des images croppées que sur chaque image croppée individuellement, afin d’éviter les cas où parmi les images croppés, une ne contiendrait pas les valeurs nutritionnelles.

Concernant le chemin, si l’algorithme détecte deux tables, alors deux images seront générées, par exemple : «950-2024-00011018\_cropped\_**0**.png» et : «950-2024-00011018\_cropped\_**1**.png». Dans le dataset, le path mentionné sera seulement le path de la dernière image générée soit ici : «950-2024-00011018\_cropped\_**1**.png».

Exemple :

|  |
| --- |
| python dataset\_llama3\_creation\_cropped.py --input\_dataset "/chemin/vers/dataset\_entree" --output\_dataset "/chemin/vers/dataset\_sortie" --output\_dir "/chemin/vers/dossier\_sortie\_images\_recadrees/" --yolo\_model "/chemin/vers/modele\_yolo.pt" –split “train” |

**Le script prend les arguments suivants :**

* ***input\_dataset*: Le chemin vers le dataset d'entrée à traiter. Ce dataset doit être au format compatible avec la bibliothèque Datasets de Hugging Face.**
* ***output\_dataset*: Le chemin où le nouveau dataset traité sera sauvegardé. Il contiendra à la fois les données originales et les nouvelles données issues de la détection de tables et de l'OCR.**
* ***output\_dir*: Le répertoire où les images recadrées (contenant les tables détectées) seront sauvegardées. Le script créera ce répertoire s'il n'existe pas.**
* ***yolo\_model*: Le chemin vers le modèle YOLO pré-entraîné pour la détection de tables. Ce modèle est utilisé pour identifier et extraire les zones de table dans les images.**

# VI)Conclusion

Je recommande fortement d'entraîner le modèle avec un dataset amélioré avec des images recadrées. Cela augmente fortement la qualité des données. De plus, il est possible de combiner la table créée en III) et une nouvelle table générée via le modèle afin de créer une nouvelle table en vérifiant uniquement les scans où il y a une non-correspondance entre les deux tables. Ceci permettrait de gagner un temps considérable pour l'élaboration d'un nouveau dataset, et ainsi atteindre les 10 000 lignes voire plus. Attention, il sera important de prendre en compte les scans anomalies car la faute viendra souvent du modèle, et ce sont justement ces types de scans que le modèle doit comprendre.