Projet AMOr : retour d'expérience sur l'entrainement d'un modèle visant la transcription automatique des formules mathématiques

Silvia Silini, Luc Bellier, Amandine Saly-Giocanti

Projet AMOr : retour d'expérience sur l'entrainement d'un modèle visant la reconnaissance automatique des formules mathématiques

Silvia Silini [a], Luc Bellier [b], Amandine Saly-Giocanti [c]

[a] Cheffe de projets numériques à la Direction des bibliothèques, de l'information et de la science ouverte de l'Université Paris-Saclay

[b] Directeur adjoint, responsable du Pôle Numérique, recherche et Science Ouverte à la Direction des bibliothèques, de l'information et de la science ouverte de l'Université Paris-Saclay

[c] Responsable de la cellule Étude et Accompagnement à la Direction des bibliothèques, de l'information et de la science ouverte de l'Université Paris-Saclay

Le projet AMOr en quelques mots

Le projet AMOr (Archives mathématiques d'Orsay) s'articule autour de deux axes :

- le premier, d'ordre archivistique, concerne le récolement et le tri des archives du Laboratoire de Mathématiques d'Orsay. Il a été mené par la Bibliothèque Jacques Hadamard.

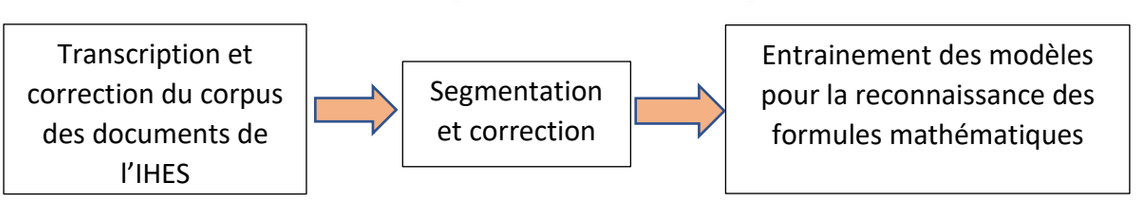
- le deuxième, relatif à l'édition des transcriptions de textes scientifiques, avait pour objectif le développement d'un modèle capable de reconnaitre les formules mathématiques présentes dans les textes pour les transcrire en LaTeX. Ces transcriptions devaient ensuite être éditées dans un éditeur XML/TEI intégré à Omeka.

Ce billet se concentrera sur la deuxième partie de ce projet, et particulièrement sur le travail exploratoire mené à la Direction des Bibliothèques et de la Science ouverte (DiBiSo) sur l'entrainement d'un modèle capable de reconnaitre les formules mathématiques. Comme corpus de vérité terrain, nous avons utilisé des documents numérisés fournis par l'IHES, le travail d'archive étant encore en cours sur ceux du Laboratoire de Mathématiques. L'échantillon utilisé comportait un grand nombre de formules mathématiques, ce qui le rendait tout à fait similaire aux documents d'archive.

Il faut savoir qu'il n'existe pas à ce jour de modèles de reconnaissance de formules librement disponibles et capables de reconnaitre des formules en plein milieu d'un document. Notre objectif était précisément de remédier à ce manque en mettant à la disposition de la communauté scientifique un modèle de ce type.

Nous avons expérimenté différentes stratégies d'entraînement de modèles, en faisant varier la quantité de vérité terrain mais aussi le type de transcription. Nous entrerons plus en détail sur ce point dans la partie dédiée à l'entrainement.

Le schéma ci-dessous représente le workflow appliqué au développement de nos modèles. L'étape de création de vérité terrain, qui consiste en la transcription des documents, ainsi que l'étape de segmentation, sont essentielles dans tous les cas tels que le nôtre, où il s'agit d'obtenir un modèle capable de reconnaitre des types d'écritures qu'il n'a jamais vus auparavant. En revanche, si on utilise un modèle de reconnaissance d'écriture déjà entrainé sur ce type d'écriture, le modèle lui-même sera capable de faire la transcription.



Transcription et correction des documents

La première étape a consisté à constituer un corpus suffisant de vérité terrain avec lequel commencer à entrainer des modèles d'apprentissage. L'objectif étant la reconnaissance des formules mathématiques dans les documents numérisés de l'IHES[1], qui seuls étaient capables de représenter une masse critique en début de projet.

La transcription de ces documents a été obtenue grâce à Mathpix, un outil propriétaire qui permet, à travers un modèle de Deep Learning (DL), d'obtenir la transcription sous différents formats (Markdown, docx, LaTeX, Overleaf, HTML, PDF) d'un texte comprenant des formules mathématiques disséminées. Nous avons fait le choix du format docx, ce qui nous a permis d'utiliser Word comme outil de correction.

Bien que Mathpix nous ait permis d'automatiser le travail de transcription, nous n'avons pas pu nous affranchir d'un contrôle de la correspondance entre chaque page transcrite et son original, afin de garantir la fidélité de la vérité terrain à sa source. En effet, Mathpix étant basé sur un modèle prédictif, les erreurs ou formules non transcrites se sont révélées fréquentes.

Au total, 302 pages ont été transcrites et corrigées. Aucun modèle n'a toutefois été entrainé avec la totalité du corpus, car nous avons privilégié une expérimentation au fur et à mesure des transcriptions et en se basant sur les résultats des modèles. A titre d'exemple, le premier modèle a été entrainé avec 195 pages.

Segmentation des documents

Une fois le contrôle de la transcription de chaque document terminé, nous avons pu procéder à leur segmentation.

L'étape de segmentation a été réalisée sur Transkribus. Cet outil a été choisi car il permet de taguer les éléments caractérisants du texte - dans notre cas les formules mathématiques. Transkribus est un logiciel propriétaire, mais accessible gratuitement. Concrètement, les modèles entrainés sont accessibles seulement à travers l'interface de Transkribus et aucune exportation de modèle n'est possible. Pour cette raison nous avons limité l'usage de Transkribus à l'étape de segmentation, et ne l'avons pas utilisé en tant que logiciel d'apprentissage des modèles.

La segmentation consiste à segmenter l'image en lignes de base (*baselines*) correspondant au niveau inférieur d'une ligne, et en régions de texte. Il s'agit concrètement des coordonnées spatiales qui identifient la présence des ces éléments à l'intérieur de la page.

Pour la segmentation, nous avons utilisé un modèle[2] déjà présent sur la plateforme qui identifie les régions de texte (contour en vert dans l'image) et les lignes de base (en bleu). Après ce premier passage automatique, un contrôle a été effectué afin de corriger les erreurs de

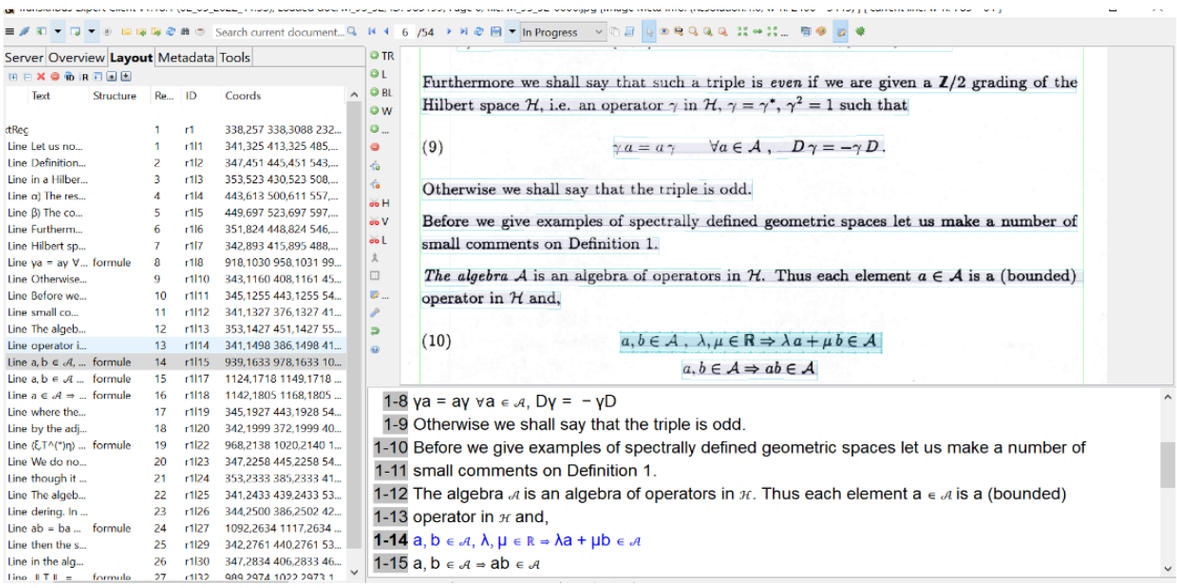
---

[1] Ici les références des documents dont a été réalisé transcription et correction :

M_74_84 (23 pages) '*Currents, flows, and diffeomorphisms*' by D. Ruelle and D.Sullivan,
M_80_06 (36 pages) '*Groups of polynomial growth and expanding maps*' by M. Gromov,
M_85_56 (24 pages) '*Averages in the plane over convex curves and maximal operators*' by J. Bourgain,
M_95_52 (54 pages) '*Noncommunicative Geometry and reality*' by A.Connes,
P_67_30 (38 pages) '*Matrices densite de polarisation*' by L. Michel,
M_79_321 (7 pages) '*An analogue of the thom isomorphism for cross products of a C* algebra by an action of* $\mathbb{R}$' by A.Connes,
M_80_04 (13 pages) '*C* Algèbres et Géométrie Différentielle*' by A.Connes,
P_78_243 (70 pages que les formules) '*The Charged Sectors of Quantum Electrodynamics in a Framework of Local Observables* ' by J.Fröhlich
M_80_28 (37 pages) '*An analogue of the thom isomorphism for cross products of a C* algebra by an action of* $\mathbb{R}$' by A.Connes.

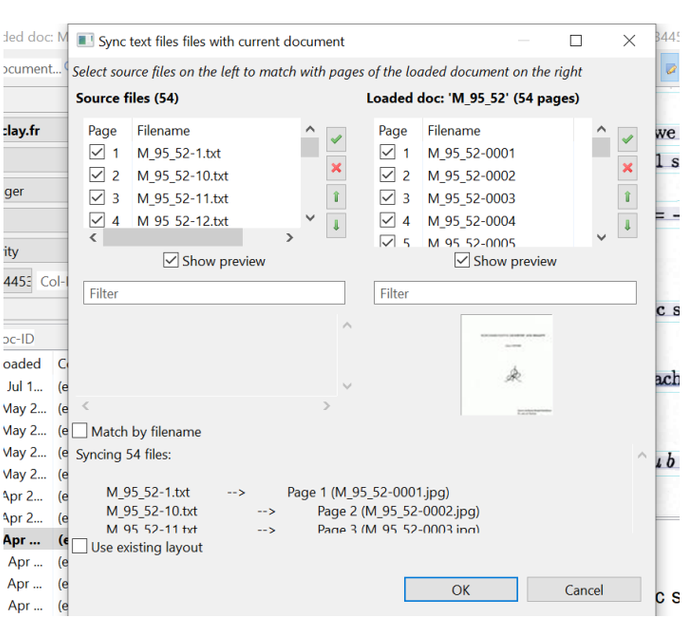
[2] CITLab Advanced

segmentation (ex. lignes de bases ne comprenant pas les indices ou les exposants) et pour ajouter un tag « *formule* » aux formules mathématiques.



Une fois la correction terminée, nous avons pu charger la transcription en la faisant correspondre avec la segmentation. Cette phase de *matching* entre coordonnées du texte dans la page et transcription correspondante est une information essentielle pour pouvoir entrainer un modèle de HTR (*handwritten texte recogni*tion) ou OCR (*optical character recognition*).

Cela a été rendu possible par une fonctionnalité de Transkribus permettant de faire correspondre la segmentation avec la transcription (capture d'écran ci-dessous). Il est important dans cette phase de synchronisation de cocher la case '*Use existing layout*' afin de garder la segmentation et la correction réalisée.



Cette fonctionnalité de *matching* automatique ne fonctionne toutefois qu'avec des transcriptions au format txt. Nous avons donc dû convertir les docx en format txt encodé en UTF8. Cela a été fait à l'aide d'un script qui convertit et supprime les retours chariot en trop dans les txt.

Une fois la correspondance entre transcription et segmentation vérifiée, nous avons pu exporter les documents depuis Transkribus vers Escriptorium, où l'entrainement des modèles a été réalisé. Ces documents ont été exportés en format XML ALTO selon la procédure présentée ici.

Apprentissage des modèles

Escriptorium est une interface web développée dans le cadre du projet SCRIPTA. On peut, comme dans Transkribus, y réaliser les taches de segmentation, transcription et d'entrainement des modèles de reconnaissance d'écriture. Mais il est open source et les modèles peuvent être exportés à l'extérieur de l'interface, raison pour laquelle il a été choisi comme outil d'entrainement des modèles à la place de Transkribus.

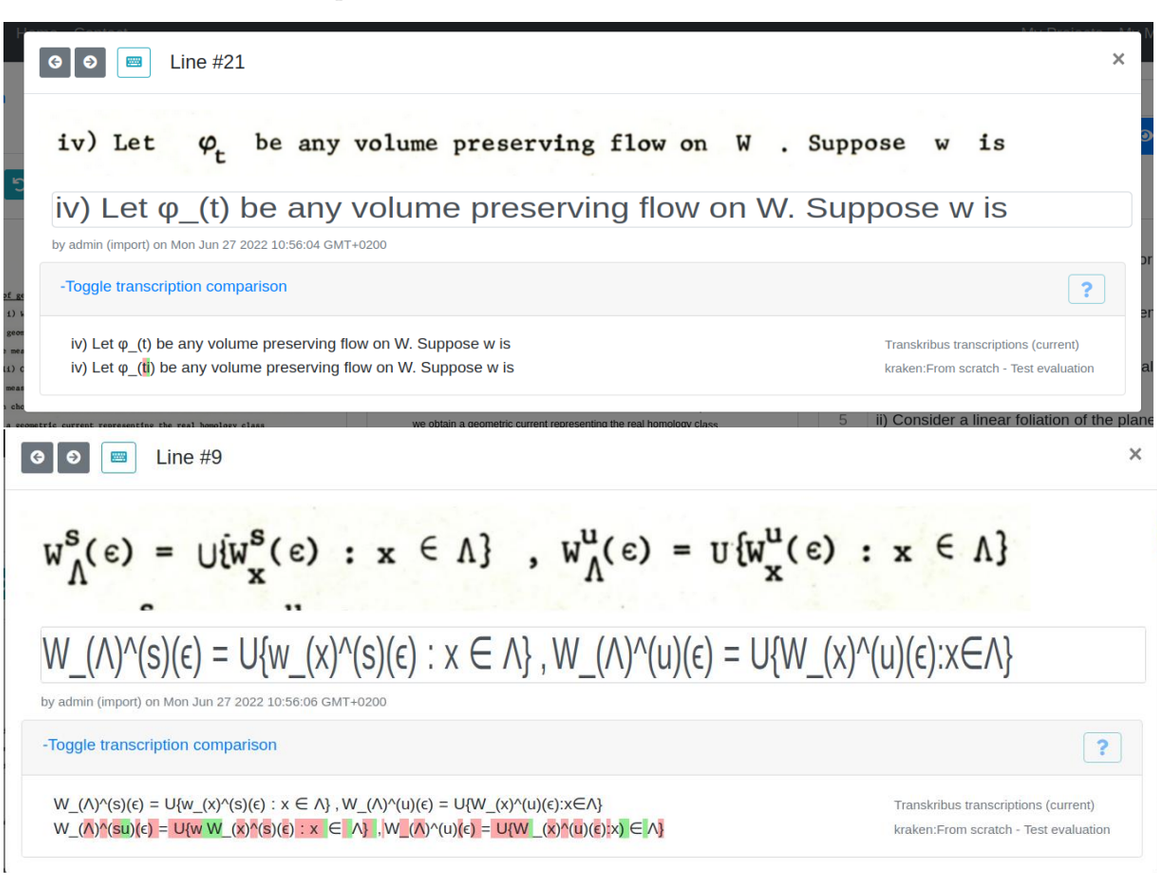
Escriptorium s'appuie actuellement sur Kraken mais vise à devenir compatible avec plusieurs solutions de transcription automatique.

Kraken est un moteur OCR en ligne de commande (CLI) qui permet la segmentation et l'entrainement des modèles pour la transcription. Nous l'avons entrainé avec les XML ALTO produits par Transkribus et les images des pages.

Le modèle à disposition dans Kraken, et donc sur Escriptorium, est une variante d'un *Variable-size Graph Specification Language (VGSL) network* qui permet la spécification d'une architecture en réseau de neurones. Nous avons utilisé le modèle de base sans changer les paramètres de l'architecture.

Une fois les images avec les transcriptions et les segmentations générées sur Transkribus chargées sur Escriptorium, nous avons commencé à entrainer des modèles de reconnaissance d'écriture. Nos premières tentatives ont été décevantes. Le texte était assez bien reconnu, mais la reconnaissance des formules, même assez simples, comportait beaucoup d'erreurs. Ce premier modèle nous a été utile pour constater que le vrai défi serait de créer un modèle capable de reconnaitre à la fois le texte et les formules.

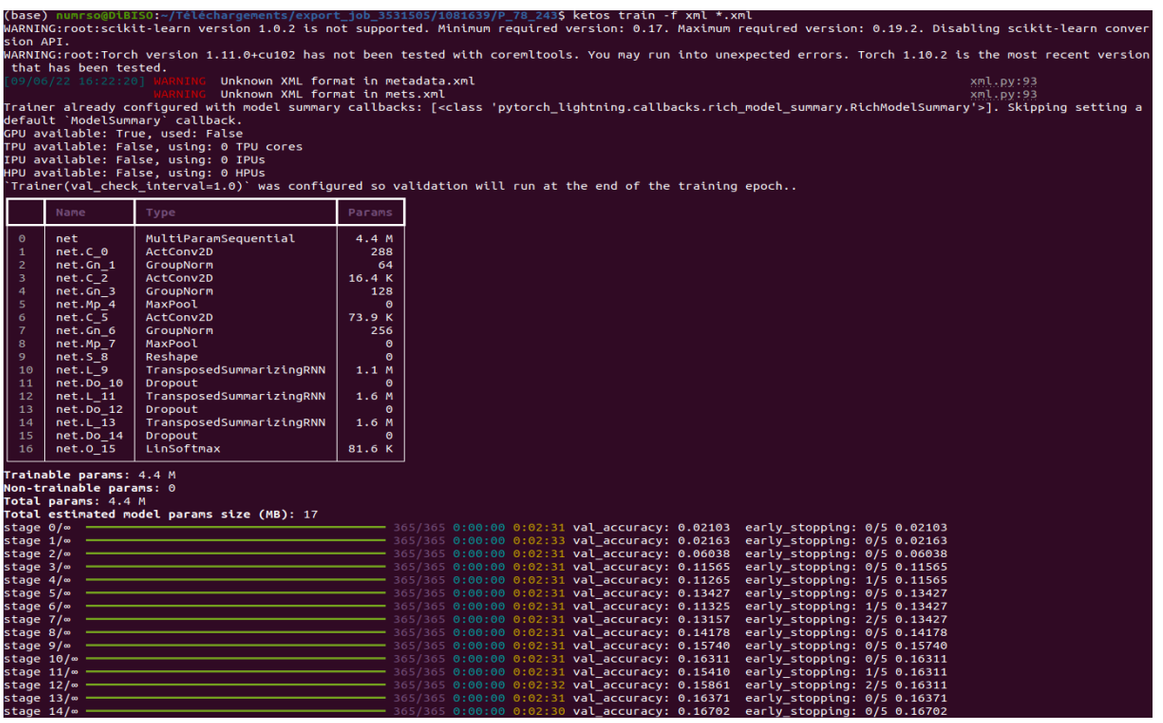
Voici deux exemples de résultats du premier modèle entrainé sur les premières 195 pages transcrites. Les couleurs représentent la différence avec la transcription Transkribus. Les caractères en vert sont ceux ajoutés par le modèle et ceux en rouge ceux supprimés mais qui devaient être normalement présents.



On constate donc que même dans le cas de formules sur une seule ligne et sans intégraux, les erreurs d'ajouts de caractères non présents ou de suppression injustifiée de caractères sont très nombreuses.

Après ce constat, nous avons donc décidé d'installer Kraken pour voir si nous pouvions nous approprier cet outil et y entrainer des modèles pour obtenir des résultats plus satisfaisants.

Pour entrainer un modèle sur Kraken, il faut d'abord placer dans un dossier les images avec les xml correspondants, en veillant à ce que chaque fichier d'image de page soit nommé de la même façon que le fichier xml contentant la transcription et la segmentation. Il faut ensuite lancer l'entrainement par ligne de commande[3] .



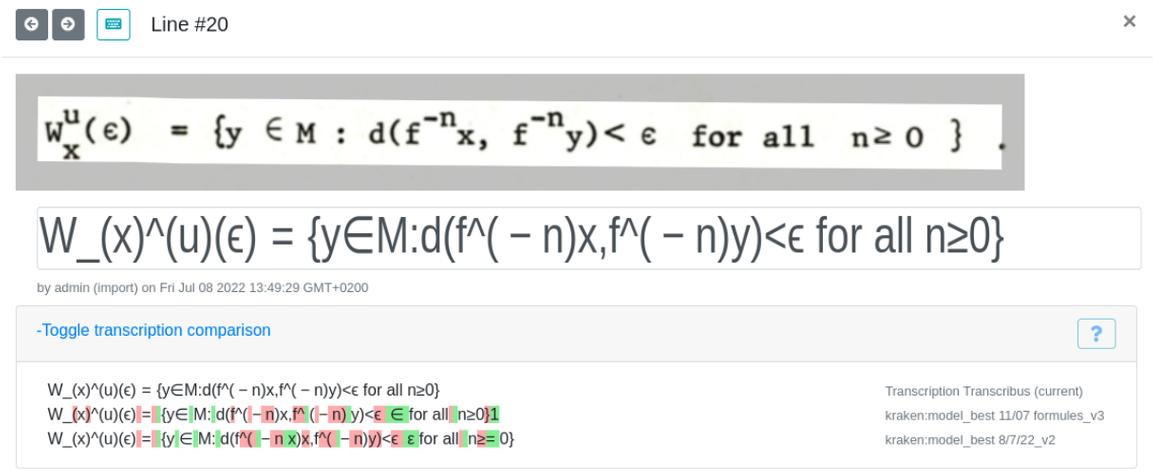
	Name	Type	Params
0	net	MultiParamSequential	4.4 M
1	net.C_0	ActConv2D	288
2	net.Gn_1	GroupNorm	64
3	net.C_2	ActConv2D	16.4 K
4	net.Gn_3	GroupNorm	128
5	net.Mp_4	MaxPool	0
6	net.C_5	ActConv2D	73.9 K
7	net.Gn_6	GroupNorm	256
8	net.Mp_7	MaxPool	0
9	net.S_8	Reshape	0
10	net.L_9	TransposedSummarizingRNN	1.1 M
11	net.Do_10	Dropout	0
12	net.L_11	TransposedSummarizingRNN	1.6 M
13	net.Do_12	Dropout	0
14	net.L_13	TransposedSummarizingRNN	1.6 M
15	net.Do_14	Dropout	0
16	net.O_15	LinSoftmax	81.6 K

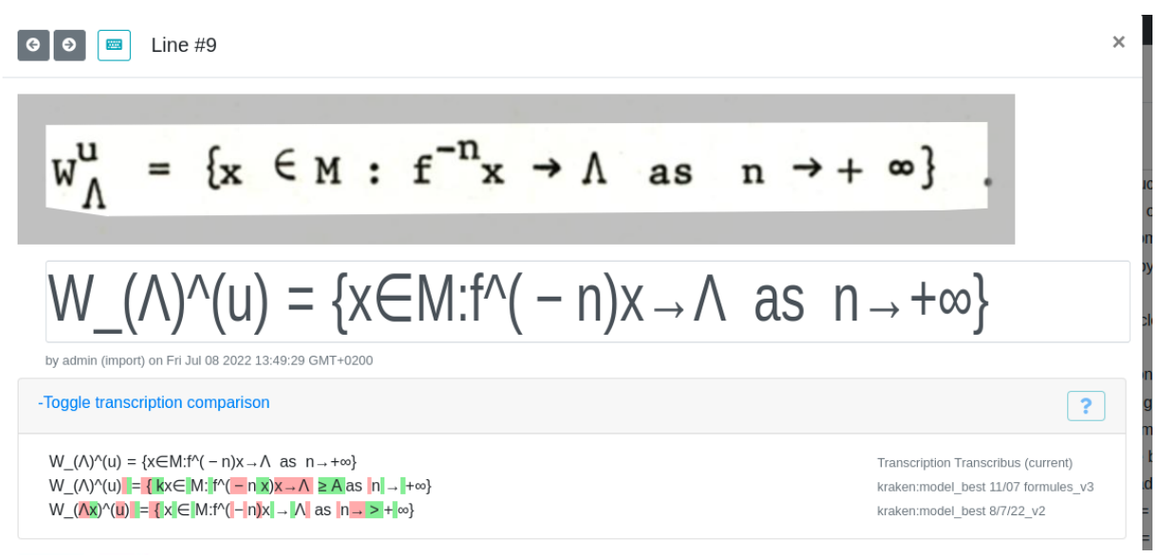
Notre première tentative a consisté à entraîner un modèle avec 23 pages (document M_74_84) pour voir si l'entrainement allait se terminer sans difficulté ni erreurs. Vu le résultat positif de ce test, nous avons entrainé un modèle avec 170 pages (M79_321, M80_04, M80_06, M85_56, M95_52, M67_30).

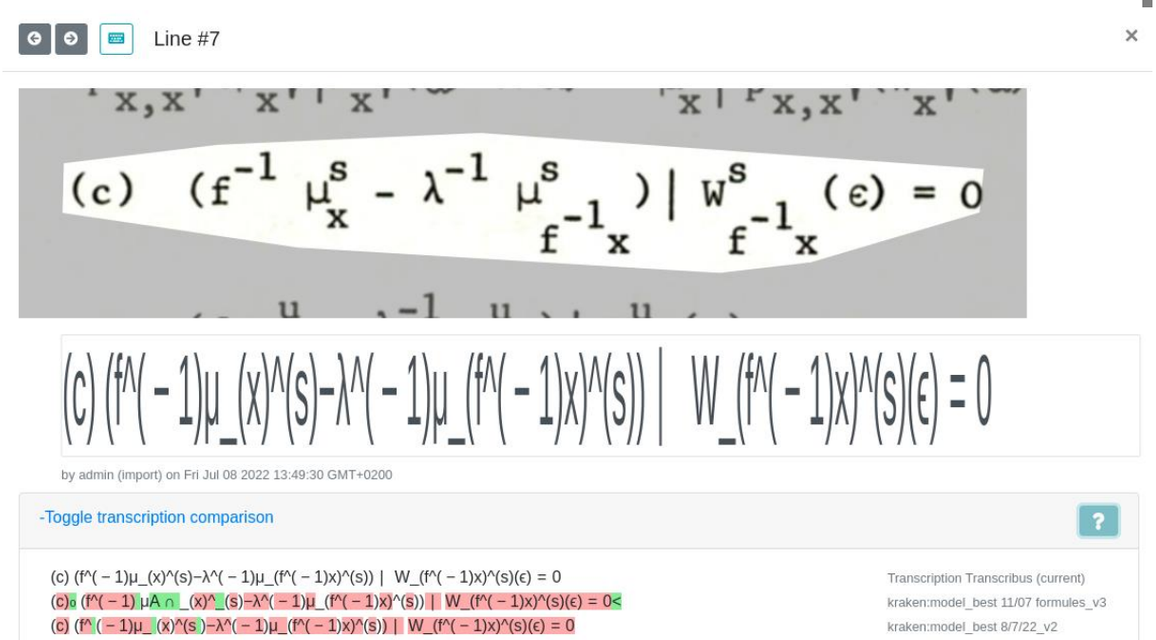
Ce premier modèle a ensuite été chargé sur Escriptorium et testé sur 143 pages. Comme il restait toujours beaucoup d'erreurs sur la reconnaissance des formules mathématiques, nous avons essayé d'entrainer un modèle avec une vérité terrain sélectionnée, c'est-à-dire d'exclure dès la vérité terrain les pages qui ne contenaient que du texte.

Cette tentative n'a pas été concluante : les formules les plus compliquées sont restées très mal reconnues et pour les autres, la reconnaissance s'est révélée très aléatoire.

[3] ketos train -f xml *.xml

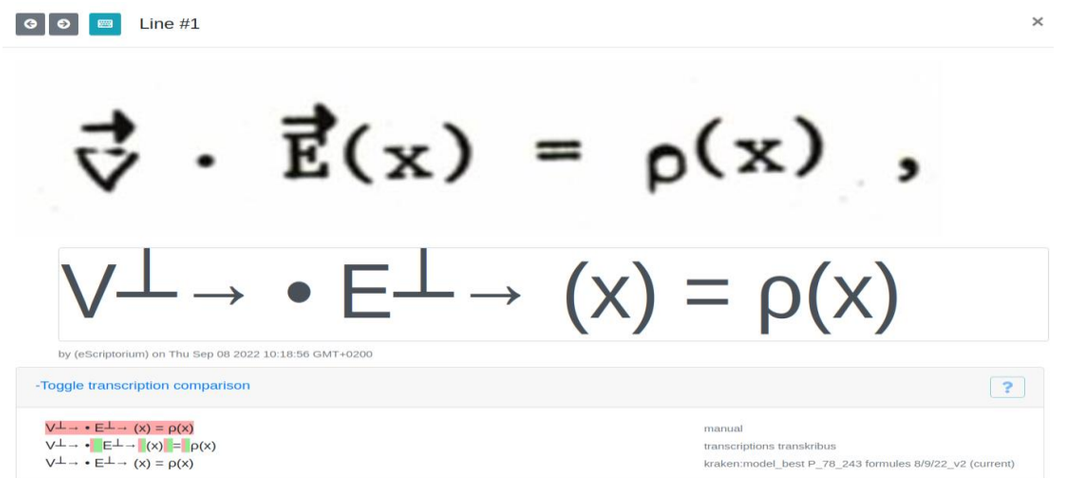


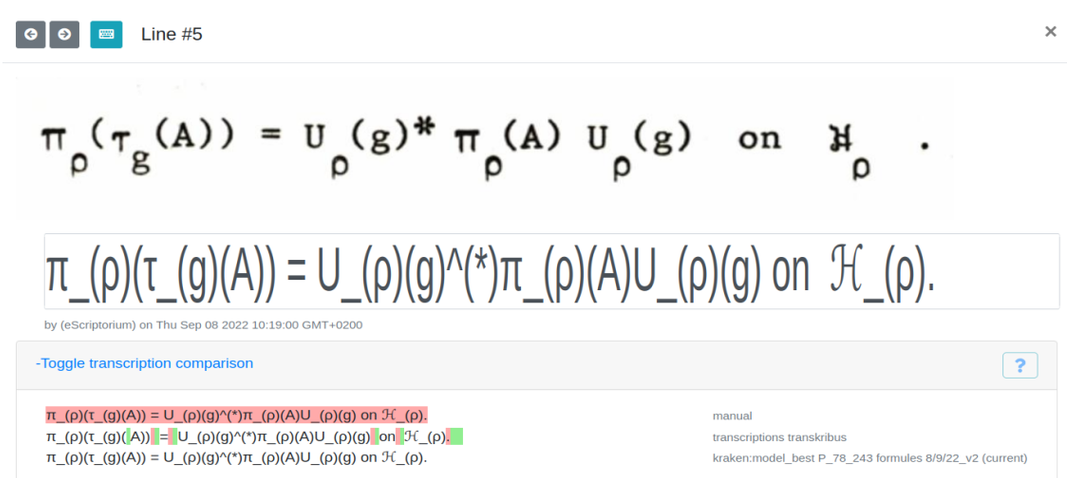




Notre dernier test a consisté à entrainer un modèle avec un document (P_78_243) constitué uniquement de transcriptions des formules mathématiques et duquel on a supprimé toutes les transcriptions de texte. Ce test reposait sur l'hypothèse que le texte provoquait du bruit et empêchait la bonne reconnaissance des formules ; le but était donc de voir si en séparant les taches (reconnaissance du texte d'abord et ensuite reconnaissance des formules), nous pourrions obtenir de meilleurs résultats sur la reconnaissance des formules. Cet essai, réalisé avec peu de vérité terrain, a été encourageant mais pas satisfaisant : le modèle a réussi à

reconnaitre des formules assez simples, mais a continué à faire beaucoup d'erreurs sur celles plus complexes, comme le montre cette capture d'écran.





Voici un tableau qui résume nos tests et les résultats obtenus :

Type de modèle	Modèle Escriptorium entrainé sur des documents contenant texte et texte+formules	Modèle Kraken entrainé sur des documents contenant texte et texte+formules	Modèle Kraken entrainé uniquement sur des pages contenant des formules	Modèle Kraken entrainé seulement sur des formules
Résultat	Bien sur le texte, décevant sur les formules	Bien sur le texte, décevant sur les formules	Bien sur le texte, décevant sur les formules	Formules sur une seule lignes bien reconnues, mais mauvaise reconnaissance des formules plus complexes

Conclusions et pistes de recherche

Bien que six mois de travail se soient révélés insuffisants pour mettre en place un modèle fonctionnel de reconnaissance des formules, nous sommes parvenus à avancer dans la transcription (302 pages des transcriptions corrigés) et à identifier de nouvelles pistes pour poursuivre le travail sur l'apprentissage automatique.

Parmi ces pistes, la modification de l'architecture du réseau de neurones de Kraken pourrait s'avérer intéressante afin d'obtenir un modèle qui parvienne à reconnaitre toutes types de formules, des plus simples aux plus complexes. Nous aurions alors une reconnaissance en deux temps :

1. reconnaissance du texte avec un modèle d'OCR classique

2. reconnaissance des formules avec notre modèle.

Une seconde piste à explorer serait de tester de nouvelles architectures, comme celle de Transformer qui semble avoir des très bons résultats dans la reconnaissance des écritures complexes.