Systèmes d'écriture et qualité des données : l'affinage de modèles de translittération dans un contexte de faibles ressources

Soumission Anonyme

RÉSUMÉ

Cet article présente une expérience visant à construire des modèles de romanisation affinés pour onze langues. Nous démontrons qu'un modèle de romanisation efficace peut être créé en affinant un modèle de base entraîné sur un corpus important d'une ou plusieurs autres langues. Le système orthographique semblerait jouer un rôle dans l'efficacité de certains modèles affinés. Nous présentons également des méthodes pour évaluer la qualité des données train et test, et comparons notre modèle arabe le plus performant à un modèle de référence.

ABSTRACT

Writing systems and data quality: finetuning transliteration models in a low-resource setting

This article presents an experiment aimed at building fine-tuned romanization models for eleven languages. We demonstrate that an effective romanization model can be created by fine-tuning a base model trained on a sizeable corpus of one or more other languages. The orthographic system appears to play a role in the effectiveness of these finetuned models. We also present methods to evaluate the quality of the train and test data, and compare our leading Arabic model against the current state of the art.

MOTS-CLÉS : Translittération automatique, langues peu dôtés, affinage.

KEYWORDS: Automatic translitteration, low-resource languages, finetuning.

1 Introduction

Le terme *romanisation* fait référence à la translittération en alphabet romain d'un texte rédigé à l'origine en écriture non-latine. La romanisation est utilisée formellement et informellement dans le monde entier à différentes fins. Par exemple, le *pinyin*, une norme de romanisation du mandarin, est largement utilisé depuis les années 1950 comme aide à la lecture aussi bien par les écoliers chinois que par les apprenants du chinois comme langue étrangère. Développé pour répondre à des besoins didactiques, ce système s'est avéré parfaitement adapté à la saisie des textes sur ordinateur des décennies plus tard. Aujourd'hui, le pinyin permet d'utiliser un simple clavier QWERTY pour rédiger des textes en chinois romanisé, qui sont ensuite convertis en sinogrammes par l'ordinateur. Avant la généralisation des smartphones proposant des claviers virtuels pour différents scripts, les arabophones s'envoyaient des SMS en *arabizi*, un système improvisé de romanisation dont la particularité est d'utiliser des chiffres pour représenter des consonnes : 3 et 7 pour ع /ʕ/ et ح /ħ/. Si les lacunes technologiques qui ont motivé la création de ce système ne sont plus d'actualité, celui-ci reste utile dans les publicités, les bandes dessinées, et les communications personnelles pour désigner un langage familier, intime, ou dialectal (Alsulami, 2019; Allehaiby, 2013). D'autres langues encore, comme le

persan, le kurde, le roumain, ou le serbo-croate, utilisent aujourd'hui différents systèmes d'écriture qui varient selon le pays, alors que certaines langues turciques de l'ancienne URSS ont utilisé trois scripts différents pendant les quelques décennies entre la révolution d'octobre 1917 et le putch d'août 1991. Ceci rend les techniques de romanisation, et de translittération plus globalement, particulièrement importantes pour rendre accessible des textes écrits sous un autre standard orthographique.

1.1 Contexte de l'étude

La présente étude se focalise sur les problématique de romanisation dans un contexte bibliothécaire. En effet, le catalogage d'un document en écriture non-latine implique généralement la production d'une romanisation fiable de son titre, de son auteur, et de sa maison d'édition. Ces translittérations servent notamment à faciliter la recherche d'un document lorsque le clavier pertinent n'est pas disponible, et à servir d'aide à la lecture pour les chercheurs et bibliothécaires qui doivent consulter le document sans forcément en maîtriser la langue. Si les alphabets grec et cyrillique peuvent être romanisés facilement en utilisant des correspondances entre caractères, la romanisation d'autres systèmes est plus difficile à automatiser. En effet, les alphabets consonantiques arabe et hébreu ne représentent pas toutes les voyelles de la majorité des langues qu'ils transcrivent, même si celles-ci sont généralement incluses dans la romanisation. Ainsi, le mot orthographique arabe de trois lettres ملك <mlk> peut être romanisé en *malik* « roi », *mulk* « royaume », *malaka* « régner » ou *mallaka* « céder ». Dans les langues sinitiques, mais aussi en japonais, les sinogrammes peuvent avoir plusieurs prononciations. Par exemple, le sinogramme 会 peut être lu comme *huì* ou bien *kuài* en mandarin selon son contexte.

Dans le réseau des bibliothèques universitaires françaises, les catalogueurs sont généralement tenus d'inclure une translittération standardisée de chaque notice bibliographique, obéissant les standards ISO ou autres normes définies par l'Agence bibliographique de l'enseignement supérieur (Abes) pour la langue en question. La production de romanisations de qualité est donc un enjeu important pour les bibliothèques multilingues de France telles que la Bibliothèque universitaire des langues et civilisations (BULAC), la Maison de l'Asie, la Contemporaine Nanterre, et la Bibliothèque de l'Institut du Monde Arabe. Cependant, les catalogues volumineux de ces bibliothèques sont susceptibles de contenir des translittérations issues de normes conflictuelles ainsi que d'erreurs de translittération. De plus, ces établissements n'ont pas toujours accès à des catalogueurs qualifiés pour identifier les erreurs ou ajouter de nouvelles romanisations pour chaque langue dans leur collection.

1.2 Translittération automatique

Nos travaux s'inscrivent dans un projet de recherche visant à répondre à ce besoin en développant des outils de translittération automatique pour les systèmes d'écriture non-latins. Si la montée des systèmes statistiques et neuronaux a permis de développer des systèmes de translittération spécifiques efficaces pour certaines langues pertinentes (ud Din, 2020; Lauc *et al.*, 2024; Davis, 2012; Terner *et al.*, 2020; Eryani & Habash, 2021), ceux-ci ne sont pas forcément compatibles avec les normes de translittération utilisées par nos bibliothèques partenaires. De plus, il existe relativement peu de ressources pour les langues moins dotées comme le pachto ou le turc ottoman, qui peuvent représenter des collections importantes pour certaines bibliothèques spécialisées. D'un point de vue plus académique, l'usage de différents architectures et de tailles de jeux de données d'entraînement rend les modèles existants difficiles à comparer. Notre problématique de translittération de langues à faibles ressources soulève

de nombres questions sur la meilleure manière de produire un modèle de translittération multilingue, dans une situation où les quantités de données disponibles sont insuffisantes pour entraîner un modèle. Notre travaille pose aussi des questions plus fondamentales sur le processus de translittération, dont de savoir si la langue ou le système d'écriture qu'une langue utilise peut avoir un impact direct sur l'efficacité d'un système de translittération neuronal. Nos premiers travaux ont montré qu'il est entièrement possible de générer de modèles de translittération efficaces pour une langue ayant seulement quelques centaines d'exemples de translittération si ce modèle est affiné sur un modèle pré-entraîné sur une autre langue ayant plus de 10 000 translittérations. De manière intéressante, nous avons constaté que l'efficacité de ces modèles affinés ne dépendait pas directement de proximités linguistiques ou orthographiques entre les langues d'affinage et les langues de base. Toutefois, cette étude était limitée à une petite variété de langues, et ne prenait pas en compte les tailles variables des jeux d'entraînement de chaque langue. La présente étude vise à approfondir nos explorations sur une plus grande diversité de langues en utilisant le même nombre de notices pour chacune.

2 Méthodologie

Nous avons basé nos travaux sur le modèle ByT5-small, une variante du modèle mT5 entraînée au niveau du byte (Xue *et al.*, 2022, 2021). Notre choix d'utiliser ce modèle plutôt qu'un modèle basé sur les tokens se justifie par sa capacité à capturer des caractéristiques morphologiques ou phonétiques encodées au niveau des caractères, ainsi que par son efficacité avérée dans des tâches similaires, comme la phonétisation (Zhu *et al.*, 2022).

Pour cette étude, nous avons exploité un corpus parallèle de notices bibliographiques en écriture originale et translittération romane. Ce corpus a été extrait du catalogue d'une grande bibliothèque multilingue de notre réseau. A partir de ce corpus, nous avons sélectionné un total de 11 langues prioritaires utilisant une écriture non-latine et comptant au moins 700 notices. Ces langues ont été choisies pour capturer une grande diversité de systèmes d'écriture et de familles linguistiques. Parmi ces langues, cinq comptaient plus de 10 000 notices. Celles-ci étaient donc les seules qui étaient adaptées à la création de modèles de base qui seraient ensuite affinés pour translittérer une langue cible.

2.1 Modèles monolingues

Pour développer nos modèles, nous avons créé cinq modèles de base en affinant byT5 sur 10 000 notices de l'arabe, du farsi, du thaï, du chinois, et du japonais sur 10 époques. Pour chacun de ces modèles, nous avons poursuivi l'entraînement pour produire onze modèles de translittération en affinant les modèles de base avec 500 notices de la langue cible sur 5 époques. Un taux d'apprentissage de 0,001 a été utilisé à chaque étape. Les onze langues cibles incluent les cinq langues utilisées pour générer les modèles de base, ainsi que six autres langues moins représentées telles que le turc ottoman ou le hindi. Une liste complète des langues et modèles peut être consulté dans la table 1.

2.2 Modèles multilingues

Nous avons également entraîné une base dite « universelle » à partir des mêmes 10 000 notices de chaque langue de base (`u10k`), pour un total de 50 000 notices dans les données d'entraînement, et une deuxième base universelle à partir de 2 000 notices de chaque langue, ou 10 000 notices en total (`u2k`). Pour chacun des 77 modèles, le premier composant du code représente le modèle de base, et le deuxième la langue utilisé pour affiner le modèle. Le modèle `per-jap` représente donc un modèle de romanisation du japonais créé avec le modèle byT5 et 10 000 notices en farsi pendant 10 époques pour le modèle de base, qui est ensuite affiné avec 500 notices en japonais. Le modèle `u2k-hin` est un modèle de translittération construit sur la base entraînée avec 2000 notices des 5 langues les plus représentées puis affiné avec 500 notices en hindi. Pour éviter que les performances des modèles soient influencées par les variations aléatoires au sein des jeux de données d'entraînement, nous avons utilisé les mêmes 10 000 notices dans l'entraînement des bases monolingues que dans la base multilingue `u10k`. Les 2000 notices de chaque langue utilisées pour entraîner la base `u2k` représentent un sous-ensemble de ces 10 000 notices, et les 500 notices utilisées pour affiner les modèles finaux sont elles-mêmes des sous-ensembles des 2000 notices utilisées dans la base `u2k`. Par conséquent, les 500 notices utilisées pour affiner le modèle `ara-ara` se trouvaient déjà parmi les 10 000 notices utilisées pour construire le modèle de base.

	Base						
Cible	**Arabe**	**Chinois**	**Farsi**	**Japonais**	**Thai**	**Uni-10k**	**Uni-2k**
Arabe	`ara-ara`	`chi-ara`	`far-ara`	`jpn-ara`	`tha-ara`	`u10k-ara`	`u2k-ara`
Chinois	`ara-chi`	`chi-chi`	`far-chi`	`jpn-chi`	`tha-chi`	`u10k-chi`	`u2k-chi`
Farsi	`ara-far`	`chi-far`	`far-far`	`jpn-far`	`tha-far`	`u10k-far`	`u2k-far`
Hébreu	`ara-heb`	`chi-heb`	`far-heb`	`jpn-heb`	`tha-heb`	`u10k-heb`	`u2k-heb`
Hindi	`ara-hin`	`chi-hin`	`far-hin`	`jpn-hin`	`tha-hin`	`u10k-hin`	`u2k-hin`
Japonais	`ara-jpn`	`chi-jpn`	`far-jpn`	`jpn-jpn`	`tha-jpn`	`u10k-jpn`	`u2k-jpn`
Turc ota.	`ara-ota`	`chi-ota`	`far-ota`	`jpn-ota`	`tha-ota`	`u10k-ota`	`u2k-ota`
Penjabi	`ara-pun`	`chi-pun`	`far-pun`	`jpn-pun`	`tha-pun`	`u10k-pun`	`u2k-pun`
Pachto	`ara-pas`	`chi-pas`	`far-pas`	`jpn-pas`	`tha-pas`	`u10k-pas`	`u2k-pas`
Thaï	`ara-tha`	`chi-tha`	`far-tha`	`jpn-tha`	`tha-tha`	`u10k-tha`	`u2k-tha`
Yiddish	`ara-yid`	`chi-yid`	`far-yid`	`jpn-yid`	`tha-yid`	`u10k-yid`	`u2k-yid`

Table 1: Modèles de translittération

Nous avons évalué chaque modèle sur un jeu de données *test* correspondant à la langue utilisée pour l'affiner. Nous avons ensuite calculé la distance de Levenshtein entre la romanisation prédite et la romanisation de référence. Étant donné la variabilité des longueurs des titres de notre corpus, et le fait que les erreurs sont plus probables d'apparaître dans les titres les plus longs, nous avons normalisé chaque distance de Levenshtein en la divisant par le nombre de caractères dans la translittération de référence. Le résultat de ce calcul est un taux d'erreur représenté comme un pourcentage de la longueur de la translittération désirée. Une translittération de 5 lettres contenant une lettre erronée aurait donc un taux d'erreur normalisé de 20%.

3 Résultats et analyse

Pour chaque modèle, nous avons calculé la moyenne des taux d'erreur normalisés. Ces moyennes sont représentées dans la carte thermique affichée dans la figure 1. L'axe Y représente les sept modèles de base, tandis que l'axe X les 11 langues avec lesquelles ces modèles ont été affinés. Les intersections représentent les moyennes des taux d'erreur pour chaque combinaison de modèle de base et de langue d'affinage.

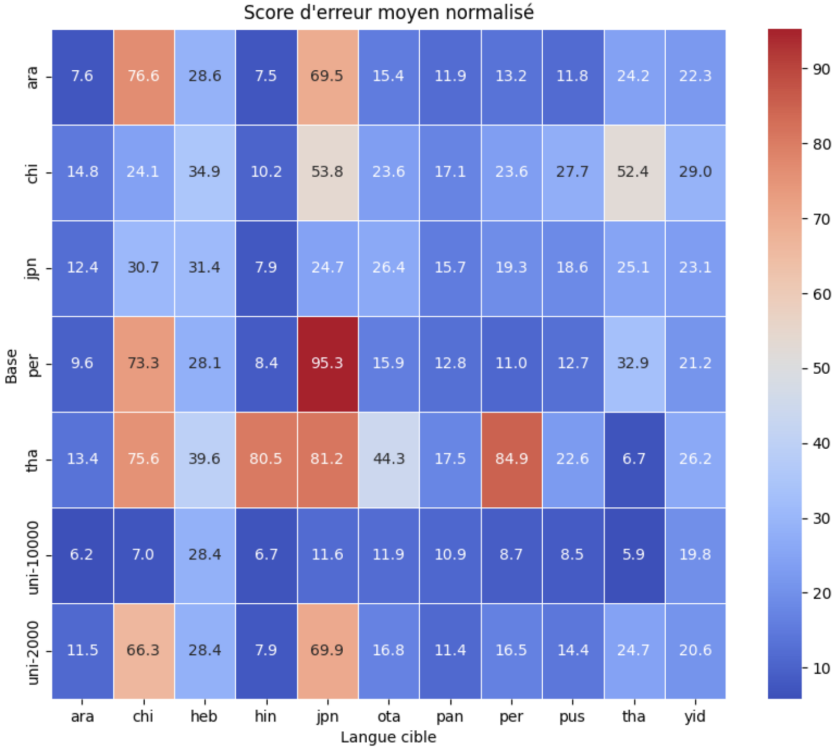

Figure 1: Carte thermique des taux d'erreurs pour chaque modèle

Nous constatons une grande variabilité en ce qui concerne l'efficacité des différents modèles, à partir de laquelle on peut tirer plusieurs conclusions intéressantes. Pour presque toutes les langues cibles, le modèle le plus efficace est celui produit en affinant la base `u10k`. Ce résultat est relativement peu surprenant pour des langues utilisant des sinogrammes (chinois, japonais) ou le script arabe (arabe, persan, pachto, penjabi, turc ottoman) car ce modèle de base a été entraîné sur plusieurs langues utilisant ces systèmes d'écriture. Cependant, ce modèle a également bénéficié au hindi, au yiddish, et dans une moindre mesure à l'hébreu dont les systèmes d'écriture ne figuraient pas dans les données utilisées pour entraîner la base.

De manière générale, nos modèles les moins performants sont ceux qui sont créés en affinant la base thaïe. Les modèles `tha-chi`, `tha-hin`, `tha-jpn`, et `tha-per` en particulier ont des taux d'erreur indicatifs qui peuvent être qualifiés d'hallucinations. Par exemple, le modèle `tha-jpn` a

translittéré presque tous les textes japonais comme *Kantō to shisei*. Si la plupart des modèles affinés sur la base `tha` ont des taux d'erreur élevés, le système d'écriture de la langue d'affinage ne semble pas jouer un rôle déterminant. Le modèle `tha-ara` est relativement performant, mais le modèle `tha-per` est parmi les moins bons produits au cours de l'expérience. Nous remarquons cependant que deux autres langues indo-iraniennes utilisant le même script, le pachto et le penjabi, ont généré des modèles relativement performants à partir de la même base. Pendant l'affinage, le thaï semble également être particulièrement incompatible avec les bases qui n'ont pas été entraînés avec des données thaïes. Si des modèles de translittération relativement performants ont pu être générés pour l'arabe et le hindi en affinant les bases japonaises et chinoises, les modèles du thaï affinés sur ces bases génèrent des taux d'erreur de plus de 25%. Pourtant, les scores pour les modèles `u10k-tha` et `tha-tha` sont excellents.

Parmi les langues d'affinage, le chinois et japonais sont également incompatibles avec la plupart des bases, comme le montre la grande proportion de rouge dans les colonnes correspondant à ces langues. De façon intéressante, si les modèles `jpn-chi` et `chi-chi` ont généré des taux d'erreur comparables (30,7% vs. 24,1%), la différence entre les modèles `jpn-jpn` (24,7%) et `chi-jpn` (53,8%) est considérable. Si on peut affiner un modèle du chinois sur du japonais, l'inverse ne semble pas être vrai. Ceci pourrait s'expliquer par la plus grande variété de prononciations possibles des sinogrammes en japonais. Il est aussi probable que le processus d'affinage ait été compliqué par le fait que le japonais utilise en plus deux syllabaires absents en chinois, le *hiragana* et le *katakana*. Il est clair que la construction de bons modèles pour ces deux langues nécessite un plus grand jeu de données en raison de la nature des systèmes d'écriture. Les 10 000 instances utilisées pour entraîner les modèles `chi-chi` et `jpn-jpn` n'ont pas été suffisants pour atteindre un taux d'erreur de moins de 24%, mais on peut supposer que les 20 000 instances contenant des sinogrammes utilisés pour entraîner la base `u10k` ont été suffisants pour atteindre des scores comparables aux modèles équivalents affinés sur des systèmes d'écriture non-logographiques.

Une dernière tendance que nous aimerions souligner est le fait que les modèles affinés sur l'hébreu et le yiddish ont généré des taux d'erreur étonnamment constants, mais jamais particulièrement bons, et ce, quelle que soit la base. De toutes les langues, se sont également celles qui ont bénéficié le moins de l'augmentation des données d'entraînement entre les modèles de base `u2k` et `u10k`. Bien que ces deux langues soient toutes les deux écrites en script hébreu, nous avons soupçonné que ces résultats particuliers sont plutôt dus à des jeux de données d'entraînement particulièrement peu fiables, comme nous confirmerons partiellement dans la partie 4.

4 Vers une évaluation humaine de nos modèles et données

4.1 Méthodes et outils d'évaluation

Dans la section précédente, nous avons évalué nos modèles sur des jeux *test* issus de notices bibliographiques pour chaque langue. Cependant, il est important de noter que les translittérations de référence dans nos jeux *train* et *test* ne sont pas forcément corrects. En effet, nos données ont été extraites directement du catalogue de notre bibliothèque partenaire sans contrôle sur la qualité des translittérations. Afin de mieux évaluer l'efficacité de nos modèles précédemment évalués par taux d'erreur ainsi que la qualité des nos jeux de données pour chaque langue, nous avons décidé d'ajouter une étape d'évaluation manuelle. Pour cela, nous avons établi une interface en ligne per-

mettant de comparer les sorties de nos meilleurs modèles aux translittérations de référence fournies par la bibliothèque. En demandant à des experts de choisir la romanisation qui représente la bonne norme, nous pouvons identifier les cas où le modèle a fourni une meilleure translittération que la référence. Pour nos futurs travaux, nous avons l'intention d'utiliser cette interface pour exclure de nos données d'entraînement les notices erronées et évaluer nos modèles avec un jeu *test* plus fiable. Nous avons exclu de l'évaluation toutes les notices pour lesquelles la prédiction a été la même que la translittération de référence.

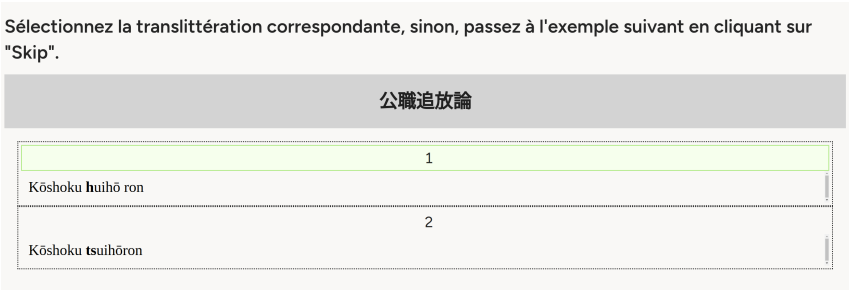

Figure 2: Interface d'annotation

Basée sur le logiciel d'annotation LabelStudio (Tkachenko *et al.*, 2020 2025), cette interface propose aux utilisateurs un choix binaire entre une translittération automatique et la translittération de référence extraite de notre catalogue, avec l'écriture originale affichée au-dessus comme nous pouvons le voir dans la figure 2. Si aucune des translittérations n'est correcte, ou s'il existe une erreur dans le texte original, l'utilisateur passe à l'exemple suivant sans choisir une des options.

Nous avons rassemblé un groupe d'annotateurs compétents de notre réseau académique et bibliothécaire pour évaluer au moins 50 exemples aléatoirement choisis pour huit des onze langues étudiées. Même s'il s'agit d'une petite proportion des notices existantes, cela nous donne une idée approximative de la fiabilité des données.

4.2 Résultats

Langue	Original (%)	Modèle (%)	Aucun (%)
Arabe	67,6	12,2	20,2
Chinois	62,5	33,9	3,6
Hébreu	44,8	24,8	30,4
Hindi	42,5	24,5	33,0
Japonais	76,2	8,5	15,3
Pachto	75,6	2,3	22,1
Penjabi	83,8	9,7	6,5
Yiddish	15,0	18,3	66,7

Table 2: Distribution des translittérations correctes entre l'originale, celle de notre modèle, ou aucune des deux

Les résultats listés dans la table 2 montrent qu'il existe de grandes disparités entre la qualité de chaque jeu de données. Nous constatons en premier lieu que notre hypothèse selon laquelle l'échantillon de notices translittérée a une qualité modérée, puisque notre système propose jusqu'à plus d'un tiers (pour le chinois) de translittérations jugées meilleures que les translittérations originales par les experts. Il faut également tenir compte du fait que ces scores ne sont pas révélateurs de la gravité des erreurs, et que les deux versions proposées peuvent parfois être très similaires : le choix de l'expert n'indique pas que la translittération originale ou du modèle n'est pas acceptable.

Pour le yiddish, 66,7% des désaccords entre le modèle et le catalogue correspondent à des situations où ni la prédiction du modèle, ni la translittération de référence n'est correcte. Ceci signifie que la majorité des notices en question sont mal translittérés, et par conséquent que le modèle apprend des correspondances incorrectes ou incohérentes à partir des translittérations existantes. Ceci peut expliquer les résultats surprenants pour cette langue dans la figure 1, où presque tous les modèles du yiddish ont généré un taux d'erreur semblable. Cependant, la distribution des réponses pour l'hébreu ne présente pas les même caractéristiques, les ensembles de données en yiddish et en hébreu ne présentent donc pas une problématique commune concernant la qualité des annotations. Enfin, notre session d'évaluation manuelle a révélé de nombreux cas où les voyelles étaient exclues dans la translittération originale, ce qui pourrait avoir contribué, au moins en partie, à des scores uniformes et généralement faibles pour l'ensemble des modèles.

En ce qui concerne d'autres langues, les experts que nous avons sollicités ont également identifié des cas où le texte original était mal transcrit. Parmi les 93 textes que nous avons annotés pour le penjabi, par exemple, 9 contenaient des fautes de frappe dans le texte original. Pour nos futurs entraînements, de tels cas seront exclus de nos jeux *test* et *train*.

Nous avons également constaté que les normes de translittération pour certaines langues sont difficiles à appliquer en pratique. C'est le cas de la nasalisation en devanagari, portée par les signes diacritiques *anusvara* et *candrabindu*, pour laquelle il existe plusieurs règles et des exceptions. Cela explique la proportion importante de translittérations rejetées par notre expert en hindi (33%). Nous notons que le modèle a malgré tout translitteré correctement 24,5% des notices. La distribution des erreurs pour cette langue suggère qu'il existe une grande proportion de translittérations erronées, mais que les erreurs sont appliquées de façon régulière.

En ce qui concerne le chinois, le modèle a réussi à prédire la bonne translittération dans 33,9% des cas, alors que seulement 3,6% des translittérations étaient erronées dans les deux cas. Ceci nous suggère que, même si le jeu de données original contient un nombre important d'erreurs, la tâche de translittération des sinogrammes étant basée sur des correspondances relativement simples en chinois, le modèle existant est capable de corriger une bonne proportion des erreurs. Nous constatons l'inverse en japonais. Cette langue a également une proportion relativement faible de cas où ni le modèle ni la notice n'ont raison. Cependant, dans les autres cas les notices sont correctes dans l'écrasante majorité des cas. On peut donc supposer que le jeu de données est relativement fiable, mais que le système orthographique est tellement complexe que le modèle n'a pas réussi à atteindre un bon niveau de précision avec si peu de données. Pendant notre séance d'annotation, l'un de nos experts du japonais nous a expliqué que les erreurs du modèle étaient "raisonnables", et que la plupart des problèmes venaient du fait que certains sinogrammes sont lus différemment s'ils font partie d'un nom propre. Vraisemblablement, notre modèle n'a pas vu suffisamment de données pour bien translittérer les noms propres.

5 Comparaison contre un modèle de référence : une étude cas avec l'arabe

Parmi les langues concernées par notre projet de recherche, l'arabe représente une priorité majeure. D'un point de vue scientifique, la sous-spécification des voyelles dans son système orthographique en fait un objet d'étude intéressant pour la translittération automatique. D'un point de vue plus pratique, le développement d'un modèle de translittération fiable aura un impact important étant donné l'importance de cette langue au sein du réseau des bibliothèques françaises. A notre connaissance, le meilleur modèle de translittération qui existe pour l'arabe est celui décrit par (Eryani & Habash, 2021). Comme notre modèle, cet outil a été développé pour translittérer des notices bibliographiques mais le fait que les données d'entraînement sont translittérés en norme ALA-LC ne correspond pas avec la norme ISO utilisée par les bibliothèques françaises. Ce modèle reste néanmoins un très bon point de référence pour comparer l'efficacité de notre propre modèle d'arabe.

En utilisant une architecture *sequence-to-sequence* basée sur les tokens, les auteurs de ce logiciels rapportent 89,7% de correspondances exacts entre les prédictions et les translittérations de référence, en ignorant les différences liées à l'emplacement des lettres majuscules. Le meilleur modèle d'arabe généré au cours de cette étude a été le modèle `u10k-ara`, qui a produit un taux d'erreur de normalisé de 6.2%. Si on évalue ce modèle en termes du nombre de matchs exacts avec les translittérations de référence, ce modèle atteint un score de 36%, loin en dessous de notre point de référence.

Malgré ce score, notre architecture et nos méthodes d'entraînement ne sont pas moins efficaces. En effet, les auteurs ont seulement atteint ce score en utilisant 480 000 notices bibliographiques comme jeu train, soit 48 fois plus de notices en arabe que notre modèle. Les mêmes auteurs ont entraîné d'autres modèles en utilisant de différentes sous-divisions de leur corpus, et la variante entraîné sur 15 000 notices d'arabe a atteint un score de seulement 28,3%. En d'autres termes, notre approche a pu atteindre un résultat significativement meilleur avec seulement deux tiers des données. Bien entendu, la base `u10k` a également été entraînée sur 40 000 notices de quatre autres langues, mais même le modèle `ara-ara` qui a *seulement* été entraîné sur 10 000 notices d'arabe a atteint un score comparable avec un taux d'erreur moyen de 7,6% et 30% de matchs exacts. Même les modèles qui ont vu seulement 500 notices d'arabe pendant l'entraînement comme `per-ara` (9,6% de taux d'erreur, 21% de matchs exacts) ont atteint des résultats prometteurs. Si nous prenons en compte les résultats de nos évaluations manuelles décrites dans la table 2, les résultats sont encore plus prometteurs. Parmi les cas où la prédiction du modèle ne correspondait pas à la translittération de la notice, le modèle avait raison dans 12,2% des cas, ce qui signifie qu'il est encore plus précis que ne le suggèrent nos métriques. Enfin, nous rappelons que nos évaluations se basent sur des translittération de référence dont seulement 67% sont jugées correctes par les experts.

Ces résultats nous font espérer que notre méthode pourrait aboutir à des résultats comparables aux meilleurs résultats de (Eryani & Habash, 2021) avec un volume moins important de données d'entraînement. Nous pensons que le modèle byT5, qui est basé sur les octets, est en grande partie responsable de cette amélioration, puisque l'architecture utilisé par (Eryani & Habash, 2021) traite chaque forme comme un token unique, excluant les informations phonétiques utiles encodées au niveau des caractères.

6 Conclusion

Dans cet article, nous présentons nos travaux visant à construire des modèles de translittération par utilisation de l'architecture byT5. Nos méthodes, qui consistent à utiliser un jeu de seulement 500 translittérations d'une langue donnée pour affiner un modèle préentraîné sur d'autres langues, donnent des résultats très prometteurs pour les langues à faibles ressources. Si la nature d'un système d'écriture semble déterminer la quantité de données nécessaires pour générer des modèles utiles pour des langues comme le chinois et le japonais, la qualité des données d'apprentissage est également un facteur important et plus complexe à maîtriser. Nous poursuivons ces travaux en intensifiant l'exploitation de l'interface d'annotation que nous avons présentée pour veiller à ce qu'une plus grande proportion de translittérations approuvées par les humains soit utilisée pour entraîner et évaluer nos modèles. Enfin, la comparaison de nos modèles avec un modèle d'arabe état de l'art suggère que, bien que nous ayons été limités par un jeu d'apprentissage beaucoup plus petit dans notre étude, nos méthodes ont le potentiel d'atteindre des performances comparables avec moins de données d'entraînement.

Références

ALLEHAIBY W. H. (2013). Arabizi: An Analysis of the Romanization of the Arabic Script from a Sociolinguistic Perspective. *Arab World English Journal*, **4**(3).

ALSULAMI A. (2019). A sociolinguistic analysis of the use of arabizi in social media among saudi arabians. *International Journal of English Linguistics*, **9**(6), 257–270.

DAVIS C. I. (2012). Tajik-Farsi Persian Transliteration Using Statistical Machine Translation. In N. CALZOLARI, K. CHOUKRI, T. DECLERCK, M. U. DOĞAN, B. MAEGAARD, J. MARIANI, A. MORENO, J. ODIJK & S. PIPERIDIS, Éds., *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, p. 3988–3995, Istanbul, Turkey: European Language Resources Association (ELRA).

ERYANI F. & HABASH N. (2021). Automatic Romanization of Arabic Bibliographic Records. In N. HABASH, H. BOUAMOR, H. HAJJ, W. MAGDY, W. ZAGHOUANI, F. BOUGARES, N. TOMEH, I. ABU FARHA & S. TOUILEB, Éds., *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, p. 213–218, Kyiv, Ukraine (Virtual): Association for Computational Linguistics.

LAUC D., RUTHERFORD A. & WONGWARAWIPATR W. (2024). AyutthayaAlpha: A Thai-Latin Script Transliteration Transformer.

TERNER O., BAR K. & DERSHOWITZ N. (2020). Transliteration of Judeo-Arabic Texts into Arabic Script using Recurrent Neural Networks. *arXiv preprint arXiv:2004.11405*.

TKACHENKO M., MALYUK M., HOLMANYUK A. & LIUBIMOV N. (2020-2025). Label Studio: Data labeling software. Open source software available at https://labelstud.io.

UD DIN U. M. (2020). Urdu-English Machine Transliteration using Neural Networks.

XUE L., BARUA A., CONSTANT N., AL-RFOU R., NARANG S., KALE M., ROBERTS A. & RAFFEL C. (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models.

XUE L., CONSTANT N., ROBERTS A., KALE M., AL-RFOU R., SIDDHANT A., BARUA A. & RAFFEL C. (2021). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In K. TOUTANOVA, A. RUMSHISKY, L. ZETTLEMOYER, D. HAKKANI-TUR, I. BELTAGY, S. BETHARD, R. COTTERELL, T. CHAKRABORTY & Y. ZHOU, Éds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 483–498, Online: Association for Computational Linguistics. DOI : 10.18653/v1/2021.naacl-main.41.

ZHU J., ZHANG C. & JURGENS D. (2022). ByT5 Model for Massively Multilingual Grapheme-to-Phoneme Conversion.