THEMATISCHE ARTIKEL

L'enseignement du latin à l'aube de l'intelligence artificielle

Des outils pour les apprenants et les enseignants

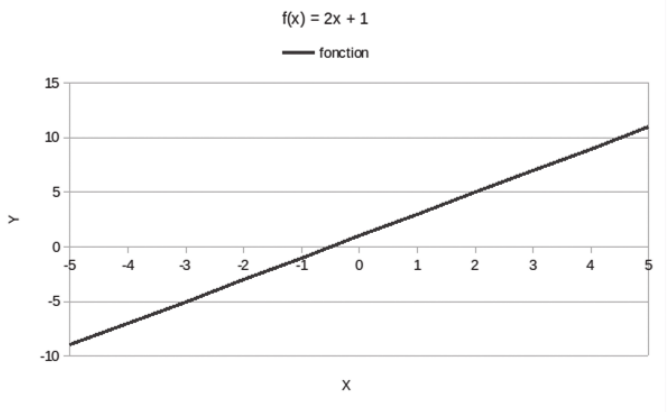
Cornelius étudie l'anglais depuis quelques mois, est capable de comprendre la majorité des textes qu'on lui donne et peut acheter un pudding sans trop de difficulté. Marie s'est dernièrement mise à l'allemand et fait des progrès fulgurants. Jacob, étudiant de latin depuis six ans, peine à traduire une phrase de Cicéron et ne sait pas s'exprimer. Quelle différence entre les trois ? Cornelius et Marie lisent des journaux dans la langue cible, écoutent des podcasts, regardent des films, parlent avec leurs amis et voyagent pendant leur temps libre, Jacob lit deux paragraphes de Tacite. Cornelius et Marie ont accès à l'immersion et à de la langue compréhensible, Jacob non. Pourquoi Jacob n'est-il pas sur un pied d'égalité avec les autres ? Parce que Jacob n'a pas accès à des locuteurs de niveau C2. Comment peut-on l'aider à en trouver ? À défaut d'en avoir sous la main, pourquoi ne chercherions-nous pas à en recréer ?

Quoique simpliste cette petite histoire résume bien l'objectif de notre projet et les raisons qui nous ont poussé à l'entreprendre. Comment, en effet, donner aux étudiants de latin des outils pour apprendre plus efficacement et de façon plus dynamique ? Notre projet consiste donc, en utilisant l'intelligence artificielle, à tenter d'offrir aux apprenants et aux enseignants des outils permettant non seulement de produire de la langue de qualité « originale » et de difficulté adaptée, mais aussi d'accompagner les apprenants dans leur processus de construction de sens. En d'autres termes, nous cherchons à faire acquérir la langue latine à une intelligence artificielle, de telle manière qu'elle puisse produire du latin compréhensible.

Commencé en 2020, le projet a rapidement pris de l'ampleur, notamment grâce au soutien de Monsieur Hersch, de Madame Kolde et de la Haute École Pédagogique vaudoise. L'outil développé est librement accessible à l'adresse *latin-ia.hepl.ch*. Dans cet article, nous présentons brièvement la nature du projet, les étapes franchies et restantes et proposons quelques pistes de réflexion et points de départ pour une application en classe ou chez soi.

Principes de l'apprentissage automatisé

« IA », « intelligence artificielle » ou encore « modèle » sont différents termes exprimant la même notion. Concrètement, une intelligence artificielle est une fonction mathématique recevant une certaine valeur et en produisant une autre. Dans un système à deux dimensions, par exemple, une fonction $f(x) = 2x + 1$ reçoit une valeur numérique appelée x et produit une valeur numérique sortante y (*i. e.* $f(x)$).



Ainsi, comme le montre la figure ci-dessus, lorsque, dans notre exemple, la valeur entrante x vaut 1, la valeur sortante y vaut 3. De la même manière, lorsque x augmente, y augmente linéairement. Nous voyons donc que la valeur sortante y est déterminée par la valeur entrante x. Connaissant la fonction $f(x) = 2x + 1$, il devient donc facile de prédire y pour n'importe quel x.

Imaginons maintenant que nous cherchions à appliquer cette même logique au langage humain. Notre objectif : créer du langage cohérent de façon automatisée.

À cet effet, nous avons déjà en notre possession le *x*, des phrases ou parties de phrase, et le *y*, la suite de ces mêmes phrases et parties de phrase. Nous connaissons, par conséquent, déjà les valeurs entrantes et sortantes de la fonction « langue ». Seule la fonction elle-même permettant de passer du *x* au *y* nous est inconnue. Or pour parvenir à codifier et à créer du langage de manière automatisée, il est nécessaire de déterminer cette fonction (pour autant qu'elle existe). C'est ici qu'entre en jeu l'intelligence artificielle, qui n'est, en fait, rien d'autre qu'un « découvreur de fonction ». Cette intelligence artificielle, au travers d'un réseau de « neurones » interconnectés, va chercher à reconstruire statistiquement la fonction recherchée. Cette phase de découverte de fonction s'appelle communément « l'entraînement ». Celui-ci suit un cheminement assez intuitif :

1. Le réseau neuronal est initialisé avec une fonction de départ (*f(x)* reçoit une structure aléatoire et de taille appropriée au problème).
2. Des valeurs entrantes *x* sont fournies au réseau neuronal.
3. Le réseau neuronal produit des valeurs sortantes y^1 comme une fonction mathématique.
4. Les valeurs sortantes y^1 sont comparées avec les valeurs sortantes connues *y* (les bonnes). Cette comparaison nous permet d'estimer « l'erreur » du système. Plus les valeurs produites y^1 sont proches des valeurs désirées *y*, plus cette erreur est faible.
5. Les variables de la fonction, appelées « poids », sont modifiées afin de réduire cette erreur.
6. Ce processus se répète en boucle jusqu'à ce que l'erreur ne puisse plus être réduite.

Sur le plan du langage, il semble que plus un réseau neuronal possède de connexions, c'est-à-dire de relations entre les neurones, plus ses prédictions sont proches d'un langage humain (pour autant que l'on dispose de suffisamment de données pour son entraînement). Le système dont il est question ici comporte 345 millions de neurones et souffre fréquemment d'incohérences. En revanche, les architectes du système, OpenAI, ont obtenu des résultats très probants en anglais avec des modèles contenant jusqu'à 175 milliards de neurones.

Le modèle

L'architecture

L'architecture du modèle est basée sur celle du GPT-2, proposée par OpenAI en 2019. GPT-2 est un réseau neuronal transformant une séquence d'unités du discours en une nouvelle unité. Cela signifie qu'il est entraîné à générer la suite d'une séquence donnée. Le modèle dont il est question dans ce projet a été entraîné sur un corpus de textes latins.

Exemple 1, génération textuelle

Déroulement :

1. Une amorce et une longueur maximale de génération sont fournies au générateur.
2. Le générateur produit un mot ou une partie de mot suivant directement l'amorce en fonction du contexte le plus probable. Il est ici important de noter que le contexte de l'amorce qu'observe le générateur existe, au moins en partie, dans le corpus.
3. Le point 2 se répète jusqu'à ce que la longueur maximale de génération ait été atteinte.

Amorce[1] : *montem uiridem ascendit*
Générateur : *[montem uiridem ascendit] et in eius cacumine sese collocauit*
Interprétation : *Il gravit une montagne verte et se plaça sur son sommet.*

L'encodage

Cherchant à traiter du langage humain, la première difficulté que nous rencontrons est celle de créer des valeurs numériques à partir des phrases latines. Nous avons vu, plus haut, que le modèle est assimilable à une fonction mathématique recevant une ou plusieurs valeurs numériques et en produisant une ou plusieurs autres. Il nous faut donc trouver un moyen pour transformer des phrases latines en nombres. Cette phase s'appelle « l'encodage ». Ici, plusieurs méthodes existent, que nous n'énumérerons pas, de peur d'ennuyer nos lecteurs. Si vous désirez en savoir plus, sachez que cette étape se nomme, en anglais, « tokenization ».

[1] Pour obtenir des productions similaires sur *latin-ia.hepl.ch*, entrez l'amorce, fixez la longueur de séquence à un maximum de 8 et le degré de fantaisie entre 5 et 7.

La méthode utilisée dans ce projet s'apparente à celle de compression par paire de bytes (BPE). Elle consiste à trouver les suites de caractères les plus fréquentes dans la langue, à les ordonner de la plus fréquente à la plus rare et à leur assigner une valeur numérique en fonction de leur position. L'ensemble de ces suites de caractères forme le « vocabulaire » du générateur. Dans ce vocabulaire, la suite de caractères la plus fréquente vaudra 1, la seconde 2, la troisième 3 et ainsi de suite jusqu'à une limite prédéfinie (50 257 dans notre projet).

Extraits du vocabulaire

Note : Le caractère « espace » est marqué « _ ». Cela signifie qu'un token « _er » correspond à une partie de mot commençant par « er », comme, par exemple, dans « **er**gastulum ». À l'inverse, un token « er » appartiendra à une partie de mot soit interne soit finale, comme, par exemple, dans « amar**er** ».

Suite de caractères	Token
_e	1
er	2
_s	3
_i	4
qu	5
um	6
...	

Concrètement, voici les diverses étapes de l'encodage.
Partons de la phrase suivante :

Quo usque tandem abutere, Catilina, patientia nostra ?

Étapes 1 et 2 : la ponctuation et les majuscules sont retirées, puis la séquence est divisée en entrées du vocabulaire.

_quo, _usque, _tandem, _abut, ere, _catilina, _patientia, _nostra

Étape 3 : Chaque entrée est remplacée par son index ou « token ».

203, 771, 2409, 13 483, 79, 39 535, 5089, 1188

Lors de l'entraînement, cette séquence est fournie au générateur. Celui-ci va tenter de découvrir une suite logique (la fonction) parmi ces nombres et produire un « token » suivant directement la séquence fournie. Ce « token » numérique est ensuite traduit, à l'aide du vocabulaire, en un mot ou partie de mot.

> 203, 771, 2409, 13 483, 79, 39 535, 5089, 1188, (token manquant : un entier entre 0 et 50 257)

Le corpus

Une fois la méthode d'encodage choisie, il s'agit de mettre en place le corpus de textes à encoder. Le « corpus » représente l'ensemble des phrases latines servant à l'apprentissage du modèle. Il est donc primordial qu'il soit non seulement immense et varié, mais surtout de haute qualité. En effet, du point de vue de l'intelligence artificielle, le corpus représente la langue latine et fait office de modèle à imiter. Nous comprenons là toute l'importance de la taille et de la variété de celui-ci. En effet, plus le réseau neuronal rencontrera de contextes d'utilisation différents pour chacun des mots du corpus, plus sa maîtrise de la langue deviendra généralisable. Dans ce contexte, « généralisable » fait référence à la capacité du modèle à généraliser ses connaissances à des contextes inconnus, c'est-à-dire, à créer des phrases correctes qui n'existent pas dans le corpus. Nous ne cherchons, en effet, pas à créer un plagieur, mais bien un « créateur » de langue. Notre objectif est de développer la capacité du modèle à généraliser ses connaissances à la langue latine.

L'entraînement

Comme nous l'avons vu ci-dessus, le modèle apprend en essayant de prédire la suite d'une séquence présente dans le corpus et en comparant sa propre génération avec la suite de la séquence du corpus. Si le modèle parvient à réduire la distance entre ces deux séquences, c'est-à-dire sa marge d'erreur, le modèle « apprend », s'il n'y parvient pas, il « plafonne » et si la marge d'erreur augmente, il « diverge ». Il est, bien entendu, dans notre intérêt que le modèle réduise l'erreur de ses propres prédictions. L'exemple suivant illustre un peu plus clairement le processus.

Le générateur reçoit une « amorce », c'est-à-dire, une séquence encodée de caractères, par exemple (Cicéron, *Oratio in L. Catilinam, 1*) :

Quo usque tandem abutere, Catilina, patientia nostra ?

Le générateur, en fonction de son analyse de la séquence, produit une suite probable :

uos (token : 777)

Cette génération est comparée avec le token suivant l'amorce dans le corpus :

quam (token : 176) *[diu etiam furor iste tuus nos eludet]*

Une distance est calculée entre le token généré (*vos*, token 777) et celui du corpus (*quam*, token 176), appelée « entropie croisée ». Le modèle cherche à minimiser cette distance. En effet, nous supposons que plus celle-ci se réduit, plus le modèle se rapproche de la fonction « langue » que nous recherchons. L'apprentissage du modèle consiste donc en une adaptation progressive de ses paramètres, au fil de ses tentatives de production et de ses résultats, en vue de minimiser son erreur. L'entraînement se poursuit de cette façon sur l'ensemble du corpus jusqu'à ce qu'il ne soit plus possible de la réduire. Un cycle d'entraînement, que nous appelons indifféremment « époque » ou « itération », correspond à une analyse de l'entier du corpus de textes. À titre de comparaison, le bref exemple mentionné ci-dessus a été réalisé après la 30^{e} itération du modèle (erreur ~ 0.05).

Cet exemple illustre également l'importance de la taille et de la variété du corpus. En effet, si, dans notre exemple, *uos* est considéré comme erroné, du point de vue de la langue latine elle-même, la suite ne l'était pas nécessairement. S'il n'existait que cette séquence de caractères dans le corpus, le modèle apprendrait simplement à rejeter *uos* après *nostra*, en d'autres termes, il « surapprendrait » en appliquant à l'ensemble de la langue latine une règle applicable seulement à une sous-catégorie minuscule (ici, une seule phrase). Évidemment, ce n'est pas ce que nous recherchons et pour minimiser ce risque deux méthodes existent :

- Augmenter la quantité des contextes de mots individuels, c'est-à-dire, augmenter la taille du corpus et le diversifier dans l'espoir d'obtenir des exemples de *uos* suivant *nostra*.

- Utiliser une méthode de validation externe non exclusive. Les expériences que nous avons menées avec le générateur dans une classe de Gymnase s'apparentent à une forme de validation externe.

L'intelligence artificielle en classe de latin

Quid des applications de l'outil ? Nous avons jusqu'ici essentiellement traité de la mise en place du système sans vraiment toucher à son utilité pratique. Dans cette section, nous passons en revue quelques exemples d'utilisation concrets et proposons quelques autres voies pour ceux que cela intéresse. Les exemples présentés ici peuvent se diviser selon le niveau de performance du générateur.

Pendant la phase d'apprentissage, le générateur fait l'objet d'analyses, d'évaluations et de corrections de la part des apprenants. Cette phase permet à ces derniers de réfléchir sur la langue ainsi que sur leurs propres pratiques et connaissances. Pour cette raison, le générateur utilisé dans ces exemples est appelé « moniteur ». En revanche, après la phase d'apprentissage, le générateur deviendrait un modèle de langue au même titre qu'un locuteur de niveau C2 et donnerait accès aux apprenants à un apprentissage par immersion et imitation. Lorsque nous faisons référence à ce type d'utilisation, nous utilisons le nom « d'auteur » pour nommer le générateur. Notez que cette dernière utilisation n'est encore qu'au stade embryonnaire et que sa réussite est largement tributaire de la qualité du corpus.

Le moniteur

Intervenant dans le processus d'acquisition d'une langue, un moniteur commet des erreurs ciblées par l'enseignant ou familières aux apprenants et reçoit un retour sur chacune de ses erreurs. Cette méthode, en introduisant un tiers dans l'environnement d'apprentissage, permet de modifier l'axe de communication et d'aider les apprenants à prendre plus facilement la parole. Ce ne sont, en effet, plus ces derniers qui sont évalués, mais bien le tiers (i.e. le générateur).

Pour que l'utilisation d'un moniteur présente un intérêt, il est primordial que les erreurs de ce dernier soient familières aux apprenants ou adaptées à leur niveau. Le générateur comporte ici l'avantage de mettre en scène des erreurs logiques ou,

du moins, facilement rationalisables. Il est, par exemple, très fréquent qu'il accorde un mot (un verbe, par exemple) avec le mot le plus proche dans la phrase, bien que le contexte montre que les deux mots n'appartiennent pas au même syntagme. De plus, il n'est pas rare qu'une erreur logique du générateur suive un raisonnement familier aux apprenants ou, du moins, qu'elle puisse être facilement rationalisée par ceux-ci.

La méthode d'application de l'intelligence artificielle en tant que « moniteur » présente plusieurs variantes. Les caractéristiques essentielles ou phases de cette approche sont : 1° l'identification des erreurs dans une séquence latine générée par le générateur (le modèle s'est-il trompé ?), 2° la correction des erreurs (qu' aurait dû faire le modèle ?) et 3° la justification des erreurs (pourquoi le modèle s'est-il trompé ?).

Du point de vue de la construction du savoir des apprenants, la phase 3° représente l'étape la plus importante du processus. En effet, c'est là que les acteurs sont amenés à explorer, verbaliser et débattre sur les concepts disciplinaires présents dans la séquence générée et sur les processus cognitifs qui auraient pu mener aux erreurs. Dans cette phase, l'enseignant devrait, dans l'idéal, ne pas prendre position, se contenter d'animer le débat et pousser les apprenants par questionnement vers des conclusions linguistiquement correctes. En effet, l'essentiel est que les apprenants soulèvent des éléments qui leur paraissent pertinents et les corrigent par eux-mêmes au travers d'une réflexion sur leurs propres processus cognitifs. Car ce sont bien les processus cognitifs familiers relevés par les apprenants et que ceux-ci attribuent au générateur qui doivent être corrigés, non ceux de l'enseignant.

Les exemples suivants illustrent la nature de la méthode dite du « moniteur ». Ces exemples sont tous tirés d'exercices réalisés tels quels dans une classe de Gymnase. Les générations avaient été amorcées avec des séquences aléatoires issues du corpus de textes ou non et les majuscules ainsi que la ponctuation avaient été ajoutées.

Exemple 2, morphosyntaxe

Générateur : Vos ista culpa in salutem rei publicae vultis admitti ?
Correction proposée : Vos ***istam culpam*** *in salutem rei publicae vultis admitti ?*

Exemple de justification : Pensant avoir affaire à une phrase simple, le modèle a mis le sujet de « admitti » au nominatif (en latin, le sujet d'une infinitive se met généralement à l'accusatif et non au nominatif).

Exemple 3, morphosyntaxe

Générateur : hocine existimas tibi fore famae si tu ad scelera confugerint ?
Correction proposée : Hocine existimas, tibi fore famae, si tu ad scelera ***confugeris*** *?*
Exemple de justification : Le modèle a cru que le verbe s'accordait avec « scelera ».

Exemple 4, cohérence

Générateur : Romani cum lacus Rhodani pontes rescidissent captivos viderunt
Correction proposée : Romani, cum ***fluminis*** *Rhodani pontes rescidissent, captivos viderunt.*
Exemple de justification : Le modèle a confondu le lac et la rivière du Rhône.

Ce type d'exercice possède plusieurs variantes et peut, par exemple, être couplé à un discriminateur pour ajouter plus de profondeur. Un discriminateur est un autre type d'intelligence artificielle discriminant une entrée selon deux catégories. Dans notre cas, le discriminateur évalue la qualité du latin en fonction de notre corpus de textes.

L'exemple suivant illustre un exercice où il s'agissait pour des élèves de latin de « battre » le générateur en créant un texte « plus authentique » (vis-à-vis du corpus) que celui-ci. Dans la première phase, la génération ainsi que son évaluation par le discriminateur étaient évaluées, puis les élèves composaient leur propre texte et comparaient leurs résultats avec ceux du générateur. Les étapes de la séquence étaient donc les suivantes :

1. Le générateur génère une séquence en fonction d'une amorce (et commet des erreurs)
2. Les élèves analysent et évaluent la séquence
3. Le discriminateur évalue la séquence générée par le générateur (et commet des erreurs)
4. Les élèves analysent et évaluent l'évaluation réalisée par le discriminateur

5. Les élèves composent une suite en fonction de la même amorce que le générateur

6. Le discriminateur évalue les compositions des élèves (et commet des erreurs)

7. Les élèves analysent et évaluent les évaluations du discriminateur

8. Les élèves observent les scores réalisés, les comparent avec ceux du générateur et les rationalisent (il s'agit ici, entre autres, de revenir sur les thèmes de la séquence et de « nouer la gerbe »)

Cette démarche offre plusieurs occasions pour les élèves de traiter des thèmes particuliers d'une leçon ou d'une séquence de façon à la fois théorique et pratique.

Exemple 5, un générateur et un discriminateur

Les phases (1-8) mentionnées ci-dessus sont indiquées entre parenthèses lorsque c'est pertinent. Notez que la taille du vocabulaire ainsi que l'algorithme d'encodage utilisés pour le discriminateur sont différents de ceux utilisés pour le générateur. Pour le discriminateur, les espaces ne sont pas considérés comme des caractères et les tokens internes ou finaux sont marqués des symboles « ## ». Les symboles [CLS] et [SEP] indiquent respectivement le début et la fin d'une séquence. Le discriminateur donne la note de 1 lorsqu'il juge que l'élément est correct, si non 0.

L'amorce (Pétrone, *Satiricon*, XII) :

> *Veniebamus in forum deficiente iam die, in quo notavimus frequentiam rerum venalium, non quidem pretiosarum sed tamen quarum fidem male ambulantem obscuritas temporis facillime tegeret. Cum ergo et ipsi raptum latrocinio pallium detulissemus, uti occasione opportunissima coepimus atque in quodam angulo laciniam extremam concutere, si quem forte emptorem splendor vestis posset adducere. Nec diu moratus rusticus quidam familiaris oculis meis cum muliercula comite propius accessit ac diligentius considerare pallium coepit. Invicem Ascyltos iniecit contemplationem super umeros rustici emptoris, ac subito exanimatus conticuit.*

La génération (1) :

> *Procedentibus deinde longius marito et muliere coactis indignatione Gitona tormentis offerentibus, ecce ancilla advenit et : 'quid ? ego, inquit, non sum digna ut collega, sed uictoris consilii sim, eiusdem etsi rei oblita ? '*

L'évaluation du discriminateur (3) : **97 %**

> [CLS] procedentibus deinde longius marito et muliere coactis
> 1 0 1 1 1 1 1 1
> indignatione git ##ona tormentis offerentibus ecce ancilla
> 1 1 1 1 1 1 1
> aduenit et quid ego inquit non sum digna ut collega sed uictoris
> 1 1 1 1 1 1 1 1 1 1 1 1 1
> consilii sim eiusdem etsi rei oblita [SEP]
> 1 1 1 1 1 1 1

L'évaluation du discriminateur pour un groupe d'élèves (6) : **100 %**

> [CLS] pallium raptum rustic ##o est ante diem ad rapi ##endum
> 1 1 1 1 1 1 1 1 1 1 1 1
> apud rusticum pallium uenimus sed mulierc ##ula nos uidit
> 1 1 1 1 1 1 1 1 1 1
> ab rustic ##o et mulierc ##ula recognoui ##m ##ur nunc
> 1 1 1 1 1 1 1 1 1 1 1
> pallium recipere uolunt [SEP]
> 1 1 1 1

L'évaluation du discriminateur pour un autre groupe d'élèves (6) : **100%**

> [CLS] asc ##yl ##tos recognosc ##ens mulierc ##ulam quam uidet
> 1 1 1 1 1 1 1 1 1 1 1
> timuit nam pallium hominis rap ##ui ##erant tamen is pallium
> 1 1 1 1 1 1 1 1 1 1 1
> recognoui ##t et dixit ubi hoc inuenisti ##s oblatum nobis est
> 1 1 1 1 1 1 1 1 1 1 1 1
> rusticus gladium cepit et sine pallio fugerunt [SEP]
> 1 1 1 1 1 1 1 1 1

Il est à noter que le discriminateur utilisé ici était lui aussi en cours d'apprentissage et que ses erreurs ont elles aussi fait l'objet d'analyses, de corrections et de justifications (7). Dans les deux exemples ci-dessus, le discriminateur n'a pas décelé les erreurs commises par les élèves (« recognovimur » et « rapuierant »), mais l'encodage permettait d'en soupçonner l'existence[2].

L'auteur

Une autre manière, actuellement hypothétique, de travailler avec le générateur est celle dite de « l'auteur ». « L'auteur » produit des documents pouvant être considérés authentiques (*i.e.* produits par un locuteur de niveau C2) et permet l'immersion au même titre qu'un locuteur d'une langue telle que l'anglais. L'authenticité dont nous parlons ici est hypothétique, utilitaire, dynamique et personnalisable. Il n'est en effet pas question de définir ce qu'est du latin « correct », mais de produire du latin similaire à ce qu'un locuteur de niveau C2 aurait pu produire. Ainsi, nous considérons ici comme « authentique » tant le latin de Plaute que celui de Tite Live, de Galilée et même d'Ørberg[3].

Les apprenants de langues modernes jouissent d'un avantage immense sur ceux des langues anciennes en ceci qu'ils ont accès à des modèles d'authenticité. Il n'est, en effet, pas difficile pour un étudiant ou un enseignant d'anglais d'accéder à du contenu adapté à son niveau dans une langue de qualité, que cela soit une personne, une vidéo, un enregistrement audio ou autre. Cet accès facilité à des modèles pratiques d'authenticité rend possible une acquisition par immersion.

À terme, notre objectif est de développer un modèle pratique d'authenticité comparable, afin de mettre la didactique du latin sur un pied d'égalité avec celle des langues modernes. En effet, si le générateur parvient à créer des séquences latines cohérentes et correctes, cela permettra aux enseignants et apprenants d'avoir accès à une source intarissable de contenu adapté à leur niveau (« comprehensible input »[4]) et, par conséquent, à une didactique par immersion.

[2] Dans ce cas particulier, le vocabulaire avait une taille de près de 120 000 entrées. Un mot composé de trois tokens ou plus pouvait généralement être considéré comme rare et, donc, erroné dans le cas d'un mot commun.

[3] Hans Henning Ørberg (1920-2010) est l'auteur de la collection *Lingua Latina per se Illustrata*. Il y propose une méthode d'apprentissage de la langue latine par la seule lecture.

[4] Selon la définition de Krashen, S. (1982). Principles and practice in second language acquisition.

D'autres applications ?

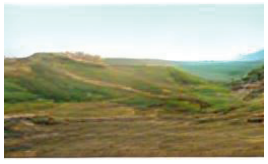
Nous nous sommes surtout attardés, dans cette section, sur des générations textuelles sans pour autant avoir été exhaustifs. Nous ne cherchions, en effet, pas à énumérer l'ensemble des possibilités d'application, de peur d'ennuyer nos lecteurs et de montrer les limites de notre propre imagination, mais plutôt à offrir quelques pistes ou points de départ. Cela ne signifie cependant pas que l'utilisation du générateur est limitée à la génération textuelle et à l'achèvement de phrases. En effet, il peut également servir à résumer un texte donné et, plus particulièrement, sous forme d'image. Dans ce cas, la séquence d'entiers générée (les « tokens ») n'est plus transformée en séquence de caractères, mais en séquence de pixels, comme l'illustrent les exemples ci-dessous. Un tel outil peut se révéler particulièrement utile pour un apprenant comme aide à la compréhension sur une expression ou un passage donné.

Comme pour la génération textuelle, le générateur d'image[5] reçoit une amorce textuelle, mais la séquence produite est une séquence de pixels (couleurs RGB). Notez que cette technologie est encore balbutiante et en phase de développement. Les résultats, bien que prometteurs, sont encore médiocres et de forme irrégulière. Les exemples suivants reprennent notre « montagne verte » mentionnée précédemment, mais, cette fois-ci, sous forme d'image.

Exemple 6, une montagne verte

Amorce : *mons viridis*

Évidemment, en traduisant soi-même le texte ou avec Google Translate, il est possible d'accéder aux modèles de génération d'images les plus récents. L'exemple suivant est une image générée à partir de la même amorce que ci-dessus, mais préalablement traduite (de façon automatisée).



[5] Le générateur d'image transforme un texte latin en une image. Il est construit selon l'architecture de DallE (OpenAI, 2021) et entraîné sur un corpus d'images et de légendes latines. Ce modèle est en cours de développement.

Exemple 7, Midjourney[6]

Amorce : *mons viridis (green mountain)*

Les langues classiques pionnières

Voilà, en quelques mots et exemples, en quoi consiste notre projet : redonner une voix à la langue latine. Nous espérons ainsi mettre à disposition des enseignants des outils pour créer des programmes vivants, permettre aux apprenants d'accéder à une didactique par immersion et rappeler aux uns comme aux autres que depuis près de trois millénaires la langue latine est la langue de l'innovation scientifique. Grâce au soutien des UER Langues et cultures et Médias, usages numériques et didactique de l'informatique de la HEP Vaud, le générateur textuel dont nous avons parlé dans cet article est librement disponible à l'adresse *latin-ia.hepl.ch*. Vous pouvez donc sans autre générer du latin depuis chez vous et tenter d'identifier les réglages vous offrant les meilleurs résultats.

La démocratisation et la prolifération des technologies liées à l'intelligence artificielle sont une chance inespérée pour la didactique des langues anciennes. Imaginez une classe de latin où l'enseignant peut obtenir à volonté, comme en anglais, en allemand, en chinois, en russe, du contenu inédit et de difficulté adéquate ! Imaginez une classe de latin où chaque élève peut facilement accéder à des textes adaptés à son niveau, à des outils visuels pour l'aider à les comprendre et à un retour détaillé en temps réel sur ses productions ! Imaginez une classe de latin où l'enseignant peut librement inviter Cicéron, Sénèque, Pline à venir donner

[6] « Midjourney » (2022) est le nom du laboratoire ayant développé, entraîné en publié le générateur d'image du même nom.

une présentation ! Profitons de l'intelligence artificielle pour redonner vie à la langue latine et montrer enfin aux élèves, aux parents, aux politiques qu'elle est non seulement une langue comme les autres, mais surtout que près de trois millénaires de locuteurs n'attendent rien d'autre que de converser avec nous !

Damien Cavaleri

Damien Cavaleri est chargé de recherche à la HEP Vaud et enseignant de Latin au secondaire II au bénéfice d'un MA en Latin et d'un MAS en enseignement. Il défend une méthode d'acquisition naturelle des langues et développe des outils gratuits pour promouvoir la langue latine. Il a créé *tutorlatin.com* afin de les rendre accessibles à tous.