Les utilisations possibles de l'Intelligence Artificielle dans la linguistique historique

Thomas HORRUT, Eliott CAMOU, Benjamin BADOUAILLE
Étudiants au Cycle Préparatoire de Bordeaux (CPBx)

Projet tutoré par Rachel Bawden

0.1 Résumé

0.2 Abstract

0.3 Remerciements

Table des matières

Table des figures

Liste des tableaux

Glossaire

distance d'édition C'est le nombre minimal d'opérations (insertion, substitution, suppression) à appliquer à une chaîne de caractères *a* pour obtenir la chaîne de caractères *b*. 15

1 Introduction

Mise en contexte pour arriver à la problématique, quel est le potentiel de l'intelligence artificielle dans la linguistique historique ? ... ?

2 La linguistique historique et l'Intelligence Artificielle

2.1 La linguistique historique

2.1.1 Introduction à la linguistique historique

Définir ce qu'est la linguistique historique, ce qu'elle étudie, et les mots de vocabulaires que nous allons rencontrer tout au long du mémoire.

La linguistique est la discipline s'intéressant à l'étude du langage. Elle se distingue de la grammaire, dans la mesure où elle n'est pas prescriptive mais descriptive. La prescription correspond à la norme, c'est-à-dire ce qui est jugé correct linguistiquement par les grammairiens. A l'inverse, la linguistique descriptive des linguistes se contente de décrire la langue telle qu'elle est et non telle qu'elle devrait être. La linguistique a des rapports très étroits avec d'autres sciences qui tantôt lui empruntent des données, tantôt lui en fournissent. Les limites qui l'en séparent n'apparaissent pas toujours nettement. Par exemple, la linguistique doit être soigneusement distinguée de l'ethnographie et de la préhistoire, où la langue n'intervient qu'à titre de document ; distinguée aussi de l'anthropologie, qui n'étudie l'homme qu'au point de vue de l'espèce. La linguistique peut se définir comme une science qui a pour objet l'étude du langage, des langues envisagées comme systèmes sous leurs aspects phonologiques, syntaxiques, lexicaux et sémantiques.

La langue est une partie du langage. C'est un ensemble de conventions nécessaires, adoptées par le corps social pour permettre l'exercice de la faculté de langage chez les individus. La langue est une convention extérieure à l'individu. Elle n'existe qu'en vertu d'une sorte de contrat passé entre les membres de la communauté. C'est une institution sociale. La langue n'est pas une fonction du sujet parlant, elle est le produit que l'individu enregistre passivement. La parole est un fait de langage distinct de la langue, c'est un acte individuel qui se distingue des institutions sociales. Elle est l'initiative d'un sujet parlant et acte de volonté et d'intelligence. Le langage quant à lui est un système de signes qui permet l'expression de la communication.

Les signes linguistiques forment un ensemble indissociable de sons et de sens, de signifiants (réalité matérielle, acoustique ou visuelle) et de signifiés (sens d'un mot). L'aspect « matériel »du signe, le signifiant, est en fait une réalité psychique : il ne s'agit pas du son comme tel, mais du son perçu. C'est pourquoi Saussure parle d' « image acoustique ». L'aspect « conceptuel »du signe, le signifié, est également une réalité psychique : il ne faut pas confondre le signifié avec le référent (ce à quoi renvoie le signe dans la réalité extérieure). Un signe a un sens (son signifié) que l'objet auquel il fait référence par ce sens existe ou non dans la réalité.

La linguistique historique permet de déchiffrer des textes anciens, mais aussi d'étudier la dynamique migratoire des humains au cours de l'histoire, grâce à des mots et des variations de langage communes. C'est une discipline de la linguistique qui étudie l'histoire et l'évolution des langues, et des familles des langues. Elle peut aussi être désignée sous le nom de linguistique comparée ou de grammaire comparée. La principale méthode de travail consiste à une comparaison entre les différents états d'une même langue ou entre des langues différentes mais issues d'un même ancêtre, mais aussi à rechercher des concordances syntaxiques ou sémantiques régulières, obéissant presque à des lois de probabilité, mettant en évidence la relation parenté entre les langues. De plus, cette comparaison et ce travail de reconstruction permet de retrouver la langue mère à partir de la langue fille. La linguistique historique permet alors de caractériser la nature des évolutions, innovations et rétentions l'état initial et les états finaux (phonétique, phonologie, lexique, syntaxe etc.)

La linguistique historique aurait été introduite par Sir William Jones (1746-1794) lorsqu'il émet l'hypothèse que le grec, le latin et le sanskrit auraient des origines communes, menant à une langue Indo-européenne. Une partie de ses hypothèses se seront révélées incorrectes plus tard, mais la langue Proto Indo-Européenne est toujours étudiée aujourd'hui, se voulant être la protoforme du latin, du gothique, du celte, du grec et du perse. Si toutes les langues indo-européennes descendent d'une langue primitive commune, quel était donc le peuple qui parlait cette langue, où se situait-il et à quelle époque ? Pour essayer de répondre à cette question, on se base généralement sur des éléments de linguistique et d'archéologie. Selon l'hypothèse kourgane, l'indo-européen viendrait d'un peuple semi-nomade ayant vécu il y a environ 6000 à 7000 ans dans la steppe située au nord de la Mer Noire, aux environs de l'actuelle frontière entre la Russie et l'Ukraine. Dans ce scénario, ce peuple de guerriers et de cavaliers conquérants aurait entrepris de nombreuses migrations, permettant ainsi la diffusion de leur langue en Europe et en Asie. Selon l'hypothèse anatolienne, l'indo-européen trouverait son origine en Anatolie (l'actuelle Turquie) il y a environ 8 à 10 000 ans, à l'époque de l'apparition de l'agriculture dans cette région. La langue se serait alors répandue dans toute l'Eurasie en même temps que la diffusion de l'agriculture. C'est actuellement l'hypothèse Kourgane qui est considérée comme la plus vraisemblable par les spécialistes, sans cependant être une certitude.

2.1.2 Les différents principes

Évidemment cette science repose sur des concepts, allant des propriétés synchoniques des mots aux à leurs aspects diachroniques.

Il existe deux manières d'aborder la linguistique historique : en étudiant une langue à un moment donné, sous un point de vue synchronique, ou en l'étudiant au cours du temps, sous l'angle diachronique. La proximité entre des langues peut traduire une histoire commune. Cependant, il faut distinguer les emprunts à une ou plusieurs autres langues des caractéristiques ancestrales de la langue étudiée. Après avoir fait la distinction, on peut dire que deux langues ayant des carac-

téristiques voisines descendent d'une langue plus ancienne commune. Par exemple, les langues romanes comme l'espagnol et le français descendent du latin.

Ainsi, à partir de ces langues voisines, on peut tenter de reconstruire leur langue ancestrale, la protolangue ou langue-mère. Selon plusieurs théories de linguistiques, cette protolangue est hypothétique et elle descendrait elle-même d'une langue plus ancienne encore. Pour reconstruire cette protolangue, on doit l'étudier sur plusieurs niveaux : la syntaxe, le lexique, la phonologie, la phonétique... De là, on peut aborder un processus récursif : à partir d'un ensemble de protolangues apparentées, on peut reconstruire une protolangue encore plus ancienne, et remonter un arbre généalogique des langues humaines de cette façon.

Pour cela, la linguistique historique doit prendre en compte le sens des mots, mais aussi leur représentation essentielle, caractérisée par une suite de sons – elle-même représentée ensuite par des symboles. Cependant, les mots de deux langues peuvent être proches, par hasard, à cause d'un emprunt, ou d'une évolution commune des langues. Par exemple, *meli* veut dire *miel* en hawaïen et en grec ancien, sans qu'un lien entre les deux langues ne soit établi. *Algorithme*, *girafe*, *orange*, sont des mots français empruntés à l'arabe. Enfin, les mots *main* et *mano* en espagnol ont la même signification et ont une origine commune. On dit alors qu'ils sont des cognats.

Un phonème est un élément sonore du langage articulé considéré d'un point de vue physiologique (disposition des organes vocaux) et d'un point de vue acoustique (perception auditive). Comme expliqué précédemment, l'évolution des langues se fait de façon plus ou moins régulière, quantifiable, et prévisible. Ces changements sont appelés correspondances régulières. Les transformations phonologiques qui changent sans exception un phonème *A* de la langue 1 en phonème *B* de la langue 2 témoignent d'une relation de parenté entre deux langues.

Un morphème est quant à lui le plus petit fragment de mot porteur de sens. Il peut être de nature lexical ou grammatical. En parallèle, on le considère également soit comme un thème morphologique, lorsqu'il porte le sens principal du mot, soit comme un affixe, dans le cas d'un préfixe ou d'un suffixe par exemple. Dans *chant-eur*, *jongl-eur*, ou *jou-eur*, le suffixe « -eur »signifie celui qui fait l'action. Ainsi, « eur »est un morphème. La Morphologie est le domaine de la linguistique étudiant les morphèmes et leur manière de composer des mots.

Ces correspondances régulières sont détectées grâce à la méthode comparative. Pour cela, elle considère un ensemble de mots. Les mots doivent alors être apparentés selon leur sens, en prenant en compte les éventuels glissements sémantiques. Après cela, les emprunts doivent être écartés. Enfin, les régularités doivent être cherchées entres les mots et les différentes évolutions qui ont pu conduire des uns aux autres n'ont plus qu'à être inspectées. (rajouter tableaux, texte explication tableaux, sources tableau)

Cependant, si des emprunts ne sont pas détectés, ils vont fausser toute mesure, en conduisant à sous-estimer la profondeur d'un ancêtre commun à plusieurs langues. Pour cela, il existe des listes de mots du lexique simple, pouvant être exploités par les linguistes, comme par exemple les listes de Swadesh (`https://en.wiktionary.org/wiki/Appendix:Latin_Swadesh_list`). Ces listes

sont utilisées en glottochronologie et en lexicostatistique, qui sont deux domaines ouverts par Morris Swadesh.

La glottochronologie, qui est la méthode de datation des langues proposée par Swadesh, fut comparée à la détermination de l'âge des fossiles à partir de la désintégration radioactive du carbone 14 (elle n'est cependant plus utilisée car sujette à beaucoup d'inexactitudes).

La lexicostatistique est une méthode utilisée en linguistique comparative et historique pour mesurer la proximité entre différentes langues. Elle se base sur l'analyse statistique des mots partagés entre ces langues, notamment les cognats, c'est-à-dire les mots ayant une origine commune. Cette approche permet d'évaluer le degré de parenté entre deux langues et de reconstituer leur histoire évolutive. En comparant la fréquence des cognats, on peut ainsi déterminer si les langues étudiées sont issues d'une même langue ancestrale. Toutefois, la lexicostatistique présente des limites, notamment en raison de possibles emprunts ou convergences culturelles qui peuvent fausser les résultats.

Cependant, bien que ces listes soient toujours utilisées, ces deux méthodes ont fait place à une application récursive de reconstruction de langue, plus adaptée et tenant compte du contexte sémantique, phonétique et syntaxique des langues.

Après une reconstruction de ces protolangues, on arrive à regrouper 12 macro-familles de langues : Ces macro-familles sont cependant aussi sujettes à débat. Bien qu'on admette comme très plausible que les proto-langues des familles de langues du monde ont fait partie à leur tour de familles encore plus anciennes, il n'y a pas d'unité de vues concernant la possibilité de reconstruire les proto-langues des superfamilles, parce qu'après une certaine période, les langues changent dans une telle mesure qu'on ne peut plus leur détecter une origine commune. (cf figure)

2.1.3 Les atouts de l'Intelligence Artificielle dans ce domaine

La linguistique historique fait face à de nombreux problèmes récurrents (traiter une grande quantité de textes pour l'homme, remarquer des motifs dans ces documents historiques). Alors que ce travail pourrait être effectué par une machine, grâce à sa capacité à traiter un grand nombres de données, et à chercher des similarités dans ces données. Avant, de voir les tâches où l'Intelligence Artificielle peut intervenir, il est d'abord nécessaire de voir en détail la conception des ces IA.

Résoudre des problèmes de Linguistique Historique avec un ordinateur nécessite de lui faire traiter du contenu textuel devant être abstrait sur des terrains parmi ceux de la **phonétique**, de la **sémantique**, de la **morphologie** ou encore de la **syntaxe**.
Développper un exemple pour illustrer ces 4 niveaux d'abstractions

La réalisation de ces abstractions s'inscrit dans le Traitement Automatisé du Langage Naturel (TAL), un domaine à cheval entre la Linguistique et l'Informatique. L'Intelligence Artificielle y occupe une place centrale pour sa capacité à effectuer des approximations améliorables avec de l'entraînement.

2.2 L'IA dans le Traitement Automatisé du Langage Naturel

2.2.1 Introduction à l'apprentissage automatique

Qu'est ce qu'une intelligence artificielle ?
Qu'est ce qu'un réseau de neurones ?
Quel est le principe derrière l'apprentissage automatique ?
Définition des apprentissages supervisés/non supervisés Définition de propagation avant. Définition rétro-propagation du gradient. Exemple de FFNN pour tâche de classification

Un important nombre de problèmes informatiques peut être résolu à travers la détermination d'une fonction mathématique f d'un espace vectoriel $\mathbb{K}^n$ vers un espace vectoriel $\mathbb{K}^{n'}$ (avec $\mathbb{K}$ correspondant à $\mathbb{R}$ ou $\mathbb{C}$).
Lorsqu'un algorithme conventionnel est développé pour réaliser un tâche, f a déjà implicitement été trouvé. Par exemple, derrière un traitement opéré sur une chaîne de caractères, elle existe bien, avec pour entrée une séquence de n caractères encodés sous forme de bits qui forme un vecteur de l'espace $\mathbb{R}^n$ et pour sortie un élément d'un espace $\mathbb{R}^{n'}$ représentant la chaîne de sortie.
En revanche, de nombreux cas demeurent où il est difficile – voire impossible – de poser une expression mathématique ou un algorithme pour répondre à certains problèmes. On considère alors f comme hypothétique et on cherche à l'approcher à partir d'un **modèle**, qu'on construit à partir des informations qu'on dispose sur f, comme un ensemble de ses points $\{(x_k, y_k = f(x_k)), k \in S\}$, à travers une tâche dite de **régression**.

Les **réseaux de neurones** sont des outils performants pour établir des modèles. Mathématiquement, ce sont des compositions d'applications non-linéaires et linéaires recevant un vecteur d'entrée représentant une donnée et sortant un vecteur de sortie représentant un résultat dans un format cohérent avec le problème.

Définitions, du neurone au réseau

Le neurone artificiel le plus élémentaire effectue la **somme pondérée** des coefficients du vecteur d'entrée, à laquelle il ajoute une valeur de **biais** pour enfin calculer l'image de la somme à travers une fonction non-linéaire dite **d'activation**. La sortie du neurone est donc un réel ou un complexe. Si on la note y_i, qu'on note x le vecteur d'entrée dans $\mathbb{K}^n$, w_i le vecteur de **poids** associé au neurone, b_i son biais et σ sa fonction d'activation, on a :

$$y_i = \sigma(b_i + \sum_{j=0}^{n} w_{ij} x_j) = \sigma(b_i + < w_i, x >) \tag{2.1}$$

[2, section 1]

Une **couche de neurones** est la mise en commun d'un nombre abritraire N de neurones devant prédire des sorties y_i différentes. Leurs vecteurs de poids w_i diffèreront donc. En revanche, leur fonction d'activation est identique. La sortie d'une couche est donc un vecteur y pouvant s'écrire

comme dans l'équation 2.2 [1].

$$
\begin{aligned}
y &= \begin{pmatrix} \sigma(b_1 + < w_1, x >) \\ \sigma(b_2 + < w_2, x >) \\ \vdots \\ \sigma(b_i + < w_i, x >) \\ \vdots \\ \sigma(b_N + < w_N, x >) \end{pmatrix} = \sigma \begin{pmatrix} b_1 + \sum_{j=0}^{n} w_{1j} x_j \\ b_2 + \sum_{j=0}^{n} w_{2j} x_j \\ \vdots \\ b_i + \sum_{j=0}^{n} w_{ij} x_j \\ \vdots \\ b_N + \sum_{j=0}^{n} w_{Nj} x_j \end{pmatrix} \\
&= \sigma \left(\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_N \end{pmatrix} + \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1j} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2j} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{i1} & w_{i2} & \dots & w_{ij} & \dots & w_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \dots & w_{Nj} & \dots & w_{Nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix} \right) = \sigma(b + Wx)
\end{aligned}
\tag{2.2}
$$

Un réseau neuronal est ainsi formé à partir de la mobilisation d'une ou plusieurs couches. L'intuition derrière l'utilisation de couches intermédiaires, qu'on nomme des **couches cachées**, est que la machine puisse être capable d'apprendre à construire des **représentations adéquates** des données pour effecuter la prédiction finalement voulue avec pertinence. On parle alors d'**apprentissage profond** et cette technique offre des réponses face aux difficultés d'abstraction soulevées par les problèmes de TAL.

L'agencement des couches et la manière de calculer la sortie finale à partir de chacune d'elles, qu'on peut rassembler sous le terme de « mode de **propagation** », est un **paramètre architecturale** à part entière qu'il faut judicieusement définir en fonction de la tâche à réaliser. Pour traiter de l'utilisation de l'IA dans le TAL, au moins deux principaux types de réseaux de neurones seront introduits au cours de ce chapitre, différant par la nature cyclique ou non de l'enchaînement de leurs couches internes.[2, introduction + section 3] [*ajouter illustration*]

Processus d'apprentissage

L'entraînement d'un réseau de neurones s'effectue à travers des **ajustements des poids et des biais dans chaque couche**, dans le cadre d'une **minimisation d'une fonction de perte** [2]. Cette fonction doit être choisie judicieusement selon le problème puisqu'elle exprime l'écart entre les sorties prédites et les sorties ciblées au passage de données d'entrée d'entraînement. Sa minimisation est réalisée par un algorithme de **descente du gradient**, avec le gradient de la fonction de perte selon tous les poids du réseau qui est calculé grâce à un procédé de **rétropropagation de l'erreur**, une adaptation pour les réseaux de neurones de celui de la discrimination rétropropagative pour des graphes d'exécution quelconques. Plusieurs implémentations de cet algorithme, qu'on nomme

1. On y confond la fonction d'activation avec la fonction vectorielle σ s'appliquant indépendamment à chaque coefficient.
2. aussi appelée fonction de coût, ou d'erreur

des **optimiseurs** [3], existent et la nature de l'optimiseur dans un modèle fait partie de ses paramètres d'entraînement.

Des hyperparamètres pour l'entraînement sont également amenés à être définis en pratique, tels que le **taux d'apprentissage**, fixant la valeur maximale de variation qu'un poids ou biais peut subir au cours d'une rétropropagation, le nombre d'**époques**, i.e. de balayages de l'ensemble des données du set d'entraînement pour l'exécution des descentes, et enfin la taille des **lots (*batchs*)**, i.e. des groupes de données d'entraînement à envoyer au modèle en une fois [4].[2, p. 17-23].[7, p. 21]

ajouter la nécessité d'initialiser aléatoirement les poids

Concevoir correctement

La phase d'entraînement du réseau s'arrête lorsque le gradient cesse de décroître.
Les hyperparamètres ainsi que certains paramètres architecturaux (nombre de couches, leur dimension) ont le mérite d'être réglés dans des configurations différentes afin de s'assurer d'une convergence optimale du gradient (qui des fois n'arrive pas ; cf. overfitting + vanishing gradient -> évoquer plus tard pour introduire les LSTM)

La phase d'évaluation du modèle se fait à partir de comparaisons des résultats sortis avec des résultats ciblés pour un set de données dit d'évaluation. Plus particulièrement en TAL, on cherche à trouver des exemples afin de déterminer les cas qu'il a su apprendre à correctement gérer ou non.
On peut également établir des métriques pour quantifier le taux de pertinence du modèle. Par exemple la distance d'édition entre les mots sortis et les mots ciblés, dans le cas de la prédiction de mots.

Quoiqu'il en soit, le choix de tous les paramètres se fait expérimentalement, à partir d'intuitions basées sur des observations lors de travaux antérieurs sur de la conception de réseaux. Pas de démonstrations mathématiques (que statistiques).
Transition vers la section suivante

Présentation du FFNN pour une tâche de classification

introduire le réseau FFNN
fonction d'activation softmax à la fin
entropie croisée pour la fonction de perte

3. Adam[8] en est un assez connu et est celui qui était prévu d'être utilisé pour le codage de l'expérience.

4. Cette quantité est pensée pour prévenir les problèmes de saturation de la mémoire des ordinateurs, qui sont favorisés par les ordres de grandeur souvent importante des tailles des sets d'entraînement.

2.2.2 Traitement des donnnées

Avant de donner à un réseau de neurones des données au format quelconque, que ce soit pour son entraînement ou pour réaliser des inférences dessus, il est nécessaire de les pré-traiter. Le but est de les rendre correctes et compréhensibles pour l'intelligence artificielle. Leur transformation se fait selon le type de tâche souhaitée. Néanmoins, dans sa généralité, les étapes de préparation des données en TAL restent les mêmes.

Tout d'abord, il faut normaliser (nettoyer) la base de données qui peut être, par exemple, un corpus de textes, une liste de mots, provenant de la toile, ou d'un système de transcription audio-visuelle. Dans ces données se trouvent des chaînes de caractères sous un certains format, où de la ponctuation peut apparaître, de même que des lettres en capitale, des chiffres, des caractères spéciaux (comme le dièse que l'on retrouve souvent dans les *tweets*)... Mais une partie de ces éléments peut ne pas avoir d'intérêt à être traitée par les algorithmes et même apporter du bruit en fournissant des données incorrectes pour entraîner l'intelligence artificielle. Une première étape est donc de s'en débarasser ou bien d'apporter des modifications à l'aide d'algorithmes de traitement de texte classique [5].[3] [Exemple]

Après avoir nettoyé la base de données, il faut définir une segmentation des textes en éléments d'entrée pertinents pour un traitement par l'IA : les *tokens*. Une tokenisation correspond à la segmentation de chaînes de caractères (comme notre texte) en tokens (mots, sous-mots, ponctuations). Par la suite, on peut extraire l'ensemble des tokens uniques d'un texte que l'on nommera : vocabulaire. Prenons un exemple « J'aime les bananes », une approche naïve est de tokeniser notre texte suivant les espaces [6] ce qui nous donne les tokens [« J'aime », « les », « bananes. »] avec un vocabulaire de taille 3 (dans cette phrase tout les tokens sont uniques). Cependant, cette approche pose des problèmes, en commençant par le token « bananes. »qui a la même signification avec ou sans point, mais qui sera considéré comme différent pour une IA. Nous pourrions ajouter la séparation suivant la ponctuation, mais alors nous obtiendrons pour « J'aime »les tokens [« J », « ' », « aime »] qui est une forme tout aussi problématique. Et il existe encore de nombreux cas (les abréviations, les points de suspension, etc.) où ce type de tokenisation pose problème. Une approche alternative est de tokeniser suivant des règles définis, par exemple, de prendre en compte les contractions comme « J'aime »et de le transformer en deux tokens [« Je », « aime »], qui est une méthode beaucoup plus efficace que la première approche, mais montrera des limites face à des situations (ou mots) rares, ou alors il faudrait spécifier de nouvelles règles pour gérer ces cas. Ainsi, la solution proposée est une approche statistique, consistant à décomposer de plus en plus un mot en sous-mots [7] au fil que sa fréquence diminue. [Exemple]. Les quatres algorithmes de tokenisation en sous-mots les plus utilisées sont le *Byte-Pair Encoding* (Sennrich et al., 2016), l'*unigram language modeling* (Kudo, 2018), le *WordPiece* (Schuster et Nakajima, 2012), et le *SentencePiece* [8] (Kudo et Richardson,

5. les Expressions Régulières sont par exemple de puissants outils pour modifier le contenu textuel

6. Vous remarquerez déjà que ce processus ne s'applique pas aux langues, comme le Japonais ou le Chinois, manquant d'espace.

7. Remarquez que le terme *mot* a un sens différent de celui qui le précède.

8. Cet algorithme est une implémentation des deux premiers.

2018).[3]

Pour la tâche de classification, vous avez vu que les mots d'entrées, [exemple], étaient convertis en une liste de nombre, autrement dit un vecteur, sur lequel il a été effectué des calculs afin d'obtenir un résultat (un nouveau vecteur), [exemple]. Une machine, un réseau de neurones, ne comprend que des nombres et ne sait procéder qu'à des calculs. Il existe différentes façons de convertir un token en un vecteur numérique, mais on retiendra deux méthodes l'encodage 1 parmi n et le plongement lexical (respectivement et plus communément appelés en anglais le *one-hot encoding* et le *word embedding*).

L'encodage 1 parmi n consiste à créer un vecteur binaire de la taille du vocabulaire $|V|$, c'est-à-dire, que chaque dimension correspond à un mot dans le vocabulaire. Dans ce vecteur binaire, la valeur est 1 pour la dimension correspondant au mot dans le vocabulaire, et 0 pour toutes les autres dimensions. Ainsi, en supposant que la dimension du mot « bananes »se trouve à la troisième dimension (c'est le troisième mot de notre vocabulaire V), sa représentation correspondra à un 1 à la troisième dimension du vecteur et à des 0 sur les autres dimensions, soit le vecteur :

$$\begin{bmatrix} 0 & 0 & 1 & 0 & \dots & 0 \\ 1 & 2 & 3 & 4 & \dots & |V| \end{bmatrix}$$

[2, (Chap 7, Jurasky)]

Comme vous pouvez le voir l'avantage de cet encodage est sa simplicité de mise en oeuvre. Mais le problème survient quand le vocabulaire devient très grand, les vecteurs générés, par définition, deviennent à leurs tours très grands et très dispersés (beaucoup de 0 et peu de 1), ce qui entraine une augmentation de la complexité du modèle (le nombre de dimension pour décrire les données) et des temps de calcul. De plus, cet encodage ne prend pas en compte les relations sémantiques entre les mots, la similarité entre deux mots ne peut être mesuré, car ils sont encodés de façon indépendante les uns des autres, ce qui peut être particulièrement problématique pour les nombreuses tâches de TAL tel que la compréhension ou traduction d'une langue.

Le plongement lexical, en revanche, permet de représenter les mots en encapsulant leur *sens sémantique* par des vecteurs denses de plus faibles dimensions, c'est à dire des vecteurs de nombres réels de petites tailles. Pour obtenir le plongement des tokens, une matrice de plongement lexical est utilisée. [9]. Cette matrice contient, à chaque ligne, le vecteur de plongement lexical d'un token du vocabulaire, et à chaque colonne correspond une dimension du vecteur de plongement. Par ailleurs, les dimensions de ces vecteurs n'ont pas une représentation claire (Jurasky, Chap 6)[5]. Cette matrice représente un espace vectoriel contenant les plongements des tokens d'un vocabulaire, ainsi, les propriétés vectoriels peuvent s'appliquer pour nos tokens vectorisés. Par exemple, étudier la similarité entre deux mots revient à calculer la distance entre les deux vecteurs correspondants, ou encore comme la figure X le montre, les vecteurs peuvent s'additionner/se soustraire permettant d'obtenir un nouveau vecteur qui garde une cohérence sémantique face à ces opérations. [Figure :

9. En réalité, les tokens sont d'abord transformer en vecteur binaire ou en un nombre correspondant à l'index dans le vocabulaire, puis la matrice de plongement lexical est appliquée.

Exemple de vecteurs sémantiques montrant bien qu'ils portent un sens et permettent des relations sémantiques entre eux ; vecteur "king" $-$ "man" $+$ "women" $->$ "queen"] Pour obtenir cette matrice de plongement lexical, deux méthodes sont possibles, soit elle est crée par l'entrainement d'un modèle personnel, soit elle est crée par un modèle externe (différent du modèle effectuant la tâche souhaitée), tel que Word2Vec, GloVe, BERT, etc.

2.2.3 Architectures neuronales utiles au TAL

Quels sont les différents outils ? Suivant, comment les parties précédentes ont été traités, ou comment les parties futures seront discutées, cette partie pourrait ne pas être nécessaire. Sinon, elle regroupera l'idée de comment on passe de notre langue naturelle à celle de la machine, de passer aux mots à des vecteurs ? Quels traitements théoriques (théoriques pour ce distinguer de la pratique dans la partie future) doient être effectués sur les mots ? En fait cette partie fait référence aux chapitres 2 et 6 de Jurasky. Voir même le chapitre 9, en supprimant la sous partie précendente pour pouvoir parler directement des réseaux de neurones appliqués à la linguistique, en d'autres termes, des réseaux récurrents, des modèles séquentielles (encodeurs-décodeurs) avec l'attention, et des Transformers.

En TAL, le besoin de pouvoir faire comprendre à l'IA des informations **contextuelles** s'est fait ressentir. Par exemple, pour étudier la langue sur le plan syntaxique, une tâche usuelle est d'attribuer à chaque mot d'une phrase sa nature grammaticale (*part-of-speech tagging*). La phrase est alors une **séquence** de *tokens* d'entrée qui sont ici des mots. On représente chacun d'eux par un *embedding* ou un vecteur one-hot. Une bonne intuition est celle que des représentations calculées en profondeur dans un réseau de neurones pour certains éléments de la séquence seront utiles pour le calcul de prédictions sur d'autres éléments de cette même séquence. De cette manière, dans la phrase « La fin de ce film n'était pas très convaincante. » et dans la phrase « Ce paragraphe n'est pas bien fin. », ce ne peut être qu'à partir de traitements sur l'ensemble de la séquence de mots qu'un modèle d'IA pourrait correctement sortir que le mot « fin »est tantôt un nom, tantôt un adjectif.

Dans cette section, des types d'architectures vont être présentés pour leur capacité à construire de manière puissante des contextes dans des séquences d'information.

Réseaux de neurones récurrents

Grâce à leur architecture cyclique, les Réseaux de Neurones Récurrents (*Recurrent Neural Networks*(RNN)) permettent de prendre en compte un contexte. Étant donné une séquence de n entrées $(x_i)_{1\leq i\leq n}$, le RNN le plus élémentaire est un enchaînement, comme dans un FFNN, d'une couche cachée avec une autre couche. La sortie est alors inférée avec l'expression :

$$y_i = \sigma(Vh_i) \tag{2.3}$$

[10] La valeur ajoutée du RNN réside dans le calcul de cette couche cachée, qui en plus d'être une fonction de l'entrée x_i de la séquence est également une fonction de la couche cachée qui a été

10. Dorénavant, les vecteurs des biais seront gardés implicites dans les expressions afin de les simplifier.

calculée en ayant passé en revue les éléments précédents de la séquence. Une somme pondérée des deux vecteurs est alors effectuée dans la définition de h_i, ce pourquoi deux matrices de poids interviennent :

$$h_i = \sigma'(Hh_{i-1} + Wx_i) \tag{2.4}$$

La propagation du flux d'informations amenant à la sortie d'un modèle RNN pour un certain élément d'une séquence d'entrée peut être illustrée avec le figuré ci-dessous :

[insérer l'illustration][4, p. 1-4]

De cette façon, on remarque aisément qu'on peut construire des empilements de couches RNN, de la même manière qu'on approfondit l'apprentissage en empilant des couches FFNN. La structure serait semblable à celle présentée ci-dessous :

[figuré montrant comment l'information + le contexte se propage à travers des empilements de couches RNN]

Dans ces schémas, la séquence d'entrées n'a été parcourue que de gauche à droite, mais il existe des problèmes, comme celui mentionné en introduction de cette section, où le contexte à gauche comme à droite d'un élément de la séquence doit être pris en compte. Une configuration architecturale des RNNs peut permettre de construire des contextes de manière *bidirectionnelle*. Dans celle-ci, la couche cachée pour l'entrée x_i est produite à partir de la concaténation des couches cachées ayant balayée respectivement les entrées $(x_1, x_2, ..., x_{i-1}, x_i)$ et dans l'autre sens les entrées $(x_n, x_{n-1}, .., x_{i+1}, x_i)$.

[Illustrer avec un schéma][4, p. 11-13]

Pour un RNN simple, 3 matrices de poids doivent donc être entraînées (en plus des biais dans chaque couche). En étalant le graphe des connexions inter-couches pour chaque traitement le long d'une séquence de données d'une certaine taille, comme illustré dans le figuré ??, il est possible d'effectuer une rétropropagation similaire à celle avec les FFNNs pour calculer les dérivées partielles de la fonction de perte selon chaque poids de ces 3 matrices.[4, p. 4, 5]

En revanche, il a été observé que la descente de gradient ne se déroule pas correctement pour des séquences de taille trop importante, ce qui empêche les simples RNNs d'exploiter efficacement une information contextuelle se situant au-delà d'un certain voisinage autour du *token* à traiter.[7, p. 25-26]

Architecture *Long Short-Term Memory*

Les *Long Short-Term Memories* sont des modèles neuronaux dont l'architecture est une variante à celle des RNNs, conçue expréssemment pour repousser les limitations de ces derniers. Pour cela, il faut entraîner le réseau de neurones à focaliser son **attention** sur les bons éléments de contexte.

Une unité LSTM est de ce fait plus complexe qu'une unité RNN, mais elle bénéficie de la même modularité. Ainsi, on peut empiler des LSTMs et construire des LSTMs bidirectionnels de la même manière que décrite plus tôt avec les RNNs.

La complexification architecturale peut dans un premier temps se remarquer d'un point de vue externe en notant que, là où une unité RNN n'est composée que d'une couche cachée qui est une

fonction de l'entrée et de la couches cachée périphérique, l'unité LSTM est composée en plus d'une **couche de contexte** qu'elle renvoie et qui est utilisée pour le calcul des couches cachées et de contexte des autres entrées dans la séquence.

D'un point de vue interne, ces deux couches interagissent à travers des **portes** rejettant et promouvant de l'information.

[Expliquer comment les portes permettent de sélectionner l'information contextuelle + transition]

Transformeurs

Précédemment, avec les RNN et les LTSM, nous avons introduit le mécanisme d'attention, permettant au réseau de se focaliser sur la manière dont les mots (éloignés) sont reliées les uns aux autres. Seulement, comme nous l'avons vu aussi, ces réseaux se basent sur des connexions récurrentes, rendant le calcul coûteux et la parallélisation difficile. Ainsi, pour pallier ce problème et gagner en performance, un nouveau modèle de réseau de neurones apparait sous le nom de *Transformers* (Transformeurs en français) dans le papier « Attention Is All You Need » de Vaswani et al. (2017)[13]. Ce modèle de type encodeur-décodeur se base sur l'attention multi-têtes, l'innovation majeure des *Transformers* [11] ou plutôt ce qui s'y trouve à l'intérieur l'auto-attention (*self-attention* en anglais).

L'attention multi-têtes permet d'étudier tous les vecteurs d'entrées, comme des mots, en même temps (de façon parallèle), et dont chaque tête qui la compose se focalise sur un aspect des *interactions* entre les différents éléments de la séquence, [exemple].

Chaque tête contient un module d'auto-attention qui est utilisé pour permettre à chaque tokens x_i de pouvoir *intéragir* avec tous les autres tokens de la séquence $X = (x_1, x_2, \ldots, x_n)$. Pour cela est extrait, pour chaque x_i, un trio de vecteurs : un vecteur requête (*query*) q_i, un vecteur clé (*key*) k_i, un vecteur valeur (*value*) v_i. Le vecteur requête correspondant au vecteur sur lequel on porte notre attention et qui sera comparé à tous les autres vecteurs, nommés les vecteurs clés. Puis, le vecteur valeur qui sera multiplié par le poids d'attention calculé avec les autres vecteurs (requêtes et clés). [Figure : Exemple de la formation des différents vecteurs q_i, k_i, v_i suivant une phrase X]. Pour créer ces vecteurs, les vecteurs x_i sont multipliés avec les matrices de poids (W^Q, W^K, W^V) qui sont obtenus à l'entrainement du modèle :

$$q_i = W^Q x_i \,;\; k_i = W^K x_i \,;\; v_i = W^V x_i \tag{2.5}$$

On pose les matrices Q, K, V respectivement l'ensemble des vecteurs q_i, k_i, v_i.

Ensuite, le calcule d'auto-attention pour une tête s'effectue par la multiplication du score d'attention (QK^T), normalisé par la racine carré de la dimension d_k de la matrice Q et K, et par la

11. Par simplification, nos efforts se concentrerons sur l'attention multi-têtes, sans évoquer qu'un *Transformers* se décompose en blocs de *transformer*, dont chaque bloc contient une unité d'attention multi-têtes (masqué ou non) et un FFNN, accompagné de connexions résiduelles et des couches de normalisation. Pour plus de détails, nous vous invitons à regarder le papier de Vaswani et al. (2017)[13].

fonction *softmax*, avec la matrice des valeurs V :

$$Attention(Q,K,V) \;=\; softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.6}$$

Enfin, la matrice d'auto-attention de toutes les têtes sont concaténées en une unique matrice qui est multipliée par une matrice de poids W^O (également obtenus à l'entrainement) pour former la matrice résultante de l'attention multi-têtes : [Figure : résumant toute les étapes.]

$$MultiHeadAttention(X) \;=\; Concat(head_1, head_2, ...)W^O \tag{2.7}$$

où $head_i = Attention(Q_i, K_i, V_i)$
avec $Q_i = XW_i^Q$; $K_i = XW_i^K$; $V_i = XW_i^V$
et i allant de 1 au nombre de têtes définis.

[Prenons un exemple : ...]

Cependant, au tout début, nous avons évoqué que le modèle étudié tout les tokens x_i en même temps, entrainant que le modèle ne tient pas compte de l'ordre des tokens, une information pourtant capitale. Par conséquent, pour injecter l'information de la position de nos tokens dans les vecteurs d'entrées, le papier Vaswani et al. (2017)[13] propose une solution en additionnant, à nos tokens plongés (vectorisés), des vecteurs positions créent à base de fonction sinus et cosinus [12]. Ainsi, cette méthode permet à notre modèle de travailler avec tous les tokens x_i, de notre séquence X, en même temps, tout en ayant l'information de leur position (dans la séquence) dans leur vecteur.

Dans certaines situations, notamment dans la tâche de modélisation de langue en TAL, où le but est de prédire le prochain mot d'une phrase, l'entrainement avec ce type de Transformeur est assez innaproprié, car vous connaissez déjà le prochain mot de votre phrase vu que vous étudiez tous les mots en même temps. Le modèle apprendra juste, à son entrainement, de renvoyer en sortie les tokens d'entrées. Pour résoudre ce problème, les chercheurs [13] propose d'appliquer un masque tel que les valeurs de la partie triangulaire supérieure de la matrice QK^T (le score d'attention) sont remplacés par des $-\infty$ (qui seront transformés en 0 par la fonction *softmax*). Cette variante se nomme l'auto-attention masquée. De ce fait, le champ de visibilité de notre modèle sera réduit au mot qu'il est en train de voir et à ceux qu'il a déjà vu, et pourra prédire de façon plus correcte (sans triche) les prochains mots de nos phrases. [Figure : montrant la matrice réduite]

12. Nous invitons le lecteur à regarder le papier de Vaswani et al. (2017)[13] pour se convaincre de la pertinence de ce choix d'encodage de position. Simplement, retenez que cela permet au modèle de s'entrainer avec la position relative des tokens, plutôt que la position absolue.

13. Ils ont proposé cette variante car leur modèle de *base* répondait aux problématiques des réseaux récurrents pour la traduction machine.

3 Les contributions de l'IA dans la linguistique historique

Petite introduction avant de passer au papiers.

3.1 Restauration de documents anciens

La restauration d'inscriptions endommagées nécessite que les épigraphistes s'appuient sur de vastes bases de données pour trouver des parallèles textuels et contextuels. Ces référentiels sont principalement constitués du répertoire mnémotechnique des parallèles d'un chercheur et, plus récemment, de corpus numériques permettant d'effectuer des recherches par « correspondance de chaînes de caractères ». Cependant, des différences dans la requête de recherche peuvent exclure ou obscurcir des résultats pertinents, et il est presque impossible d'estimer la véritable distribution de probabilité des restaurations possibles. L'attribution d'une inscription est tout aussi problématique : si elle a été déplacée ou si des éléments de datation internes utiles manquent, les historiens doivent trouver d'autres critères pour attribuer le lieu et la date de l'écriture (formes de lettres, dialectes, etc). Et cela implique inévitablement, un niveau élevé de généralisation, les intervalles d'attribution chronologique pouvant être très longs.

Ainsi, avec l'utilisation d'Ithaca, nous surmontons les contraintes des méthodes épigraphiques actuelles en utilisant le deep learning, un apprentissage automatique. Les réseaux neuronaux profonds peuvent découvrir et exploiter des modèles statistiques complexes dans de grandes quantités de données. L'augmentation récente de la puissance de calcul a permis à ces modèles de relever des défis de plus en plus sophistiqués dans de nombreux domaines, y compris l'étude des langues anciennes. Ithaca, possède une architecture de réseau neuronal profond (*deep neural network* en anglais) entraînée à effectuer simultanément les tâches de restauration textuelle, d'attribution géographique et d'attribution chronologique. Ithaca, a été formé sur des inscriptions écrites en grec ancien et dans le monde méditerranéen entre le VIIe et le XXe siècle. Ce choix s'explique par deux raisons principales. Premièrement, la variabilité du contenu et du contexte des documents épigraphiques grecs, qui en fait un excellent défi pour le traitement du langage ; et deuxièmement, la disponibilité de corpus numérisés pour le grec ancien, une ressource essentielle pour l'entraînement des modèles d'apprentissage automatique.

L'architecture d'Ithaca a été soigneusement adaptée à chacune des trois tâches épigraphiques, en traitant de manière pertinente les informations contextuelles à long terme et en produisant des résultats interprétables pour améliorer la synergie entre homme et machine. Pour commencer, les informations contextuelles sont capturées de manière plus complète en représentant les entrées

sous forme de mots ; cependant, des parties de mots ont pu être perdues au cours des siècles. Pour relever ce défi, nous traitons le texte d'entrée sous forme de représentations de caractères et de mots conjointement, en représentant les mots endommagés, manquants ou inconnus (unknown) par un symbole spécial « [unk] ».

Ensuite, pour permettre un traitement à grande échelle, Ithaca est basé sur une architecture de type Transformeur, qui utilise le mécanisme d'attention pour évaluer l'influence des différentes parties de l'entrée (telles que les caractères, les mots) sur le processus de prise de décision du modèle. Le mécanisme d'attention est informé de la position de chaque partie du texte d'entrée en concaténant les représentations des caractères et des mots d'entrée avec leurs informations positionnelles séquentielles. Ithaca est constitué de blocs de transformeur empilés : chaque bloc produit une séquence de représentations traitées dont la longueur est égale au nombre de caractères d'entrée, et la sortie de chaque bloc devient l'entrée du suivant. La sortie finale de l'ensemble des blocs est transmise à trois têtes [1] de tâches différentes qui traitent respectivement la restauration, l'attribution géographique et l'attribution chronologique. Chaque tête est constituée d'un réseau neuronal feedforward peu profond, spécifiquement entraîné pour chaque tâche.

Ithaca est conçu pour assister et étendre le travail de l'historien. L'architecture d'Ithaca est axée sur la collaboration, l'aide à la décision et l'interprétabilité. Alors que Ithaca seul atteint une précision de 62% lors de la restauration de textes endommagés, l'utilisation d'Ithaca par des historiens a amélioré leur précision de 25% à 72%, confirmant l'effet synergique de cet outil de recherche. Ithaca peut de plus attribuer des inscriptions à leur emplacement d'origine avec une précision de 71% et peut les dater à moins de 30 ans de leur période de vérité, redonnant ainsi vie à des textes clés de l'Athènes classique et contribuant à des débats d'actualité sur l'histoire ancienne. Cette recherche montre comment des modèles tels qu'Ithaca peuvent libérer le potentiel de coopération entre l'intelligence artificielle et les historiens, en transformant la façon dont nous étudions et écrivons sur l'une des périodes les plus importantes de l'histoire de l'humanité.

Ithaca est ainsi un exemple type de l'utilisation des intelligences artificielles dans le domaine de la linguistique historique. Elle n'a pour l'instant été développée qu'avec des langues connues et déchiffrées, mais l'un des enjeux des chercheurs est d'étendre l'action des intelligences artificielles jusqu'au déchiffrement de langues anciennes, indéchiffrables à ce jour. La plupart de ces langues sont issues de langues mortes, n'ayant pas de locuteurs ni suffisamment de supports physiques permettant son étude approfondie. Le linear B, une écriture mycénienne du deuxième millénaire av. J.-C., a été découvert en Crète en 1900 et compris seulement en 1952, tandis que le linear A, découvert au même moment, n'a toujours pas été déchiffré à l'heure actuelle. On estime qu'à l'heure actuelle, une douzaine de langues sont toujours indéchiffrées. Plusieurs projets sont d'ailleurs en cours de développement, exploitant les données récoltées depuis le début de l'utilisation des intelligences artificielles.

1. Attention ces têtes sont différentes de celles contenus dans les blocs de transformeur, les multi-têtes d'attention.

3.2 Déchiffrement de langues anciennes

Le prochain défi des intelligences artificielles est le déchiffrement de langues anciennes. Pour déchiffrer une langue ancienne, l'objectif des linguistes et archéologues est de trouver des textes appelés des « bilingues », qui permettent à la fois de comparer ces deux langues, et de traduire la langue inconnue de la même manière que Champollion en 1821. Plus de 200 ans après la découverte de la pierre de Rosette, on peut maintenant affirmer que Champollion a pu déchiffrer l'égyptien autrement que grâce à cette dernière. Elle a en effet servi à amorcer la compréhension du système alphabétique de l'égyptien, mais Champollion a surtout déchiffré l'égyptien grâce au copte, la langue liturgique des chrétiens d'Egypte. L'apport du copte a permis 95 % du déchiffrement, la pierre de Rosette - ou ses équivalents - à peine 5%. Cependant, ces bilingues sont très rares et parfois trompeurs, car rédigés différemment ou erronés dans leur traduction, et les linguistes doivent s'appuyer sur d'autres textes et méthodes. Tout d'abord, les linguistes doivent déterminer le type de signe utilisé : une écriture alphabétique ou alpha-syllabique, une écriture qui note chaque syllabe indépendamment, ou alors un système qui comporte des logogrammes c'est-à-dire des signes utilisés pour noter des mots entiers.

L'autre méthode consiste à travailler à partir des noms propres, comme a pu le faire récemment François Desset pour déchiffrer l'élamite linéaire. C'est grâce à huit vases en argent que l'archéologue français est parvenu à *craquer* le code. Ces huit vases, nommés *vases Gunagis*, vieux de 4 000 ans, présentent des séquences de signes identiques d'une pièce à l'autre. L'archéologue a ainsi pu repérer les noms de deux souverains, Shilhaha et Ebarti II, ainsi que de la divinité Napirisha, probablement représentée par un serpent, qui l'ont petit à petit amené à déchiffrer le reste de l'écriture en élamite linéaire [6].

Plusieurs travaux sur le déchiffrement de langues anciennes avec de l'intelligence artificielle ont été mené, notamment par le MIT en octobre 2020. Les chercheurs du Computer Science and Artificial Intelligence Laboratory (CSAIL) ont réalisé une avancée majeure dans ce domaine [11] : ils ont mis au point un nouveau système capable de déchiffrer automatiquement une langue perdue, sans avoir besoin d'une connaissance *approfondie* de ses relations avec d'autres langues. Les chercheurs ont également montré que leur système peut lui-même déterminer les relations entre les langues et l'ont utilisé pour corroborer des travaux récents suggérant que la langue ibérique n'est pas réellement liée au basque [1].

Le système repose sur plusieurs principes issus de la linguistique historique, comme le fait que les langues n'évoluent généralement que de certaines manières prévisibles. Par exemple, s'il est rare qu'une langue donnée ajoute ou supprime un son entier, certaines substitutions de sons sont susceptibles de se produire plutôt que d'autres. Un mot comportant un « p »dans la langue mère peut se transformer en « b »dans la langue descendante, mais il est moins probable qu'il se transforme en « k »en raison de l'écart important entre les prononciations. L'algorithme apprend à intégrer les sons de la langue dans un espace multidimensionnel où les différences de prononciation se reflètent dans la distance entre les vecteurs correspondants. Cette conception leur permet de capturer des modèles pertinents de changement linguistique et de les exprimer sous forme de

contraintes informatiques. Le modèle qui en résulte peut segmenter les mots d'une langue ancienne et les mettre en correspondance avec leurs équivalents dans une langue apparentée.

Cependant, cet algorithme est uniquement basé sur des caractères phonétiques de l'alphabet international phonétique (IPA). Les valeurs phonétiques des caractères perdus peuvent être déduites en les mettant en correspondance avec les caractères apparentés connus et ces valeurs peuvent servir comme point de départ pour la reconstruction des sons perdus, et des recherches plus approfondies sont nécessaires pour établir leur efficacité.

L'article abordé propose un modèle de déchiffrement pour extraire les cognats des textes non-segmentés (ou sous-segmentés), sans supposer une proximité entre les langues perdues et les langues connues, comparé à leur papier précédent[10]. Les propriétés linguistiques de chaque langue ont été incorporées dans la conception du modèle, telles que la plausibilité phonétique du changement de son et la préservation des sons. Les résultats de l'étude sur le gothique, l'ougaritique et l'ibérique montrent que leur modèle peut traiter efficacement les textes sous-segmentés, même lorsque les langues source et cible ne sont pas apparentées. En outre, ils introduisent une méthode d'identification des langues proches qui trouve correctement les langues apparentées pour le gothique et l'ougaritique. Pour l'ibérique, la méthode n'apporte pas de preuves solides en faveur du basque comme langue apparentée, ce qui correspond à la position privilégiée par les chercheurs actuels [1]. Les applications potentielles de leur méthode ne se limitent pas au déchiffrement. Les valeurs phonétiques des caractères perdus peuvent être déduites en les mettant en correspondance avec les caractères apparentés connus. Ces valeurs peuvent servir de point de départ à la reconstruction des sons perdus et des recherches supplémentaires sont nécessaires pour établir leur efficacité. En outre, l'efficacité de l'intégration des caractéristiques phonologiques ouvre la voie à de futures améliorations pour la détection des cognats dans la linguistique historique computationnelle. Actuellement, la méthode fonctionne sur une paire de langues. Pour traiter simultanément plusieurs langues, comme c'est souvent le cas dans la tâche de détection de cognats, des travaux supplémentaires sont nécessaires pour modifier ce modèle actuel et sa procédure d'inférence.

4 Étude du cas de l'application de l'IA pour la reconstruction des proto-formes

La méthode comparative décrite en section 2.1.2 permet de comprendre le sens et la prononciation de mots dans une langue disparue à partir de comparaisons avec d'autres langues apparentées dont on dispose de plus de connaissances. Ce problème a de l'importance pour la construction de lexiques et pour la traduction de documents dans des langues déjà déchiffrées. Le cas de l'Étrusque l'illustre bien puisque, bien que déchiffré (des informations sur la phonétique et la syntaxe ont été trouvées), les linguistes ignorent encore à quelles langues le comparer pour comprendre le sens d'un grand nombre de ses mots.[9].

Des solutions informatiques ont été développées pour appliquer cette méthode-là dans des directions différentes. Par exemple, l'identification des cognats dans des langues aux origines communes est une tâche pour laquelle de nombreux travaux de recherche ont été entrepris et à l'issue desquels de performantes solutions sont sorties.[7][12, 2 : Related Works] Cependant, notre projet s'est focalisé sur celle de la reconstruction des proto-formes, dont de récentes solutions neuronales sont apparues. Puisqu'une d'entre-elles a fait l'objet de la rédaction de notre article scientifique et de la mise en place d'une expérience, cet exemple d'application de l'IA va être étudié en détail dans ce chapitre.

4.1 État de l'art

4.1.1 Conceptualisation du problème

Définir clairement le problème du titre, énoncer et justifier le choix de notre modèle réseaux de neurones et des différents outils appliqués. Voir s'il est possible de faire apparaitre plusieurs démarches, c'est à dire, une approche statistique et une approche neuronale (toujours pour renforcer et montrer le potentiel de l'IA).

Le problème que cette tâche doit résoudre est de prédire la proto-forme la plus probable d'ascendre à un ensemble donné de cognats. Les indices à suivre pour effectuer la reconstruction sont ici les règles de variations phonétiques de la linguistique diachronique.
Puisque ces variations sont supposées régulières, il est possible de déployer des réseaux de neurones pour les apprendre à partir d'un grand nombre de données, qui sont des paires composées d'un groupe de cognats dans des langues sœurs et de la proto-forme supposée leut être correctement associée. Toutes les chaînes de caractères sont converties sous forme phonétique à partir de l'Alphabet Phonétique International (*International Phonetic Alphabet* (IPA)) afin que les tokens à manipuler ne soient que des caractères phonétiques, représentant des sons.

4.1.2 Dernières solutions neuronales

Approche supervisée

En disposant des données décrites plus tôt, la dernière démarche supervisée développée ces dernières années est directement inspirée de celles mobilisées usuellement dans les problèmes de **traduction automatique**. Le modèle neuronal est alors doté de l'architecture **encodeur-décodeur**.

L'encodeur reçoit les séquences d'*embeddings* associés aux caractères de chaque cognats et encode les informations qui vont être passées dans le décodeur pour générer une proto-forme plausible. Les deux parties sont chacun des réseaux de neurones à part, mais chacune sont en réalité des RNNs composés d'unités GRU.

Approche non-supervisée

4.1.3 Limites d'applicabilité

[Expliquer en quoi le non-supervisé donne plus d'espoir que le supervisé mais en quoi même cette approche présente des limites.]
[Transition avec la problématique de l'article scientifique]

4.2 Expérience sur une approche non-superivsée

4.2.1 Méthode

4.2.2 Mise en place

4.2.3 Analyse

4.2.4 Critiques

5 Conclusion

5.1 Synthèse

Résume tout ce qui a été dit.

5.2 Les différentes limites posées aujourd'hui

une partie des limites aura déjà été traitée dans le chapitre précédent. Cette sous partie se veut résumer ces limites, et aller dans les limites générales (voir acutelles) de l'IA dans la linguisitique historique.

5.3 Les perspectives de l'IA dans la linguistique historique

Ouverture, dépassement de certaines limites, évolution des modèles.

Bibliographie

[1] Yannis ASSAEL et al. « Restoring and attributing ancient texts using deep neural networks ». In : *Nature* 603.7900 (mars 2022), p. 280-283. ISSN : 0028-0836. DOI : 10.1038/s41586-022-04448-z. URL : https://europepmc.org/articles/PMC8907065.

[2] James H. Martin DAN JURAFSKY. « Neural Networks and Neural Language Models ». In : *Speech and Language Processing*. 2023. Chap. 7. URL : https://web.stanford.edu/~jurafsky/slp3/7.

[3] James H. Martin DAN JURAFSKY. « Regular Expressions, Text Normalization, Edit Distance ». In : *Speech and Language Processing*. 2023. Chap. 2. URL : https://web.stanford.edu/~jurafsky/slp3/2.

[4] James H. Martin DAN JURAFSKY. « RNNs and LSTMs ». In : *Speech and Language Processing*. Chap. 9. URL : https://web.stanford.edu/~jurafsky/slp3/7.

[5] James H. Martin DAN JURAFSKY. « Vector Semantics and Embeddings ». In : *Speech and Language Processing*. 2023. Chap. 6. URL : https://web.stanford.edu/~jurafsky/slp3/6.

[6] François DESSET et al. « The Decipherment of Linear Elamite Writing ». In : *Zeitschrift für Assyriologie und vorderasiatische Archäologie* 112.1 (2022), p. 11-60. DOI : doi:10.1515/za-2022-0003. URL : https://doi.org/10.1515/za-2022-0003.

[7] Clémentine FOURRIER. « Neural Approaches to Historical Word Reconstruction ». Theses. Université PSL (Paris Sciences & Lettres), sept. 2022. URL : https://hal.inria.fr/tel-03793299.

[8] Diederik P. KINGMA et Jimmy BA. *Adam : A Method for Stochastic Optimization*. 2017. arXiv : 1412.6980 [cs.LG].

[9] *L'écriture et la langue étrusques : histoire d'un déchiffrement et d'une conquête scientifique en cours*. BNF. Mars 2022. URL : https://www.bnf.fr/fr/agenda/lecriture-et-la-langue-etrusques-histoire-dun-dechiffrement-et-dune-conquete-scientifique-en.

[10] Jiaming LUO, Yuan CAO et Regina BARZILAY. « Neural Decipherment via Minimum-Cost Flow : from Ugaritic to Linear B ». In : *CoRR* abs/1906.06718 (2019). arXiv : 1906.06718. URL : http://arxiv.org/abs/1906.06718.

[11] Jiaming LUO et al. « Deciphering Undersegmented Ancient Scripts Using Phonetic Prior ». In : *CoRR* abs/2010.11054 (2020). arXiv : 2010.11054. URL : https://arxiv.org/abs/2010.11054.

[12] Carlo MELONI, Shauli RAVFOGEL et Yoav GOLDBERG. « Ab Antiquo : Neural Proto-language Reconstruction ». In : *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*. Online : Association for Computational Linguistics, juin 2021, p. 4460-4473. DOI : 10.18653/v1/2021.naacl-main.353. URL : https://aclanthology.org/2021.naacl-main.353.

[13] Ashish VASWANI et al. *Attention Is All You Need*. 2017. arXiv : 1706.03762 [cs.CL].