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

3 étudiants de CPBx

0.1 Résumé

0.2 Abstract

0.3 Remerciements

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

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

Chapitre 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.

2.1.2 Les différents principes

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

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.
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 :

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

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].

1. On y confond la fonction d'activation avec la fonction vectorielle σ s'appliquant indépendamment à chaque coefficient.

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

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.[1]

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 comprendre l'utilisation de l'IA dans le TAL, au moins deux principaux types de réseaux de neurones devront être introduits au cours de ce chapitre, différant par la nature cyclique ou non de l'enchaînement des couches internes.

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 de fonctions de perte**, déterminée par les sorties temporairement prédites par le réseau.

dire que les informations dans chaque coeff du vecteur x sont souvent abstraites et n'ont de sens que pour la machine (teaser vers sous- section suivante) + leur importance est déterminée par les poids avec l'entraînement
décrire l'entraînement (régression logistique+fct perte)
Exemple de la tâche de classification (connotation textuelle)
dire que σ est un paramètre du réseau
on peut empiler plusieurs réseaux de neurones et plusieurs couches (paramètre architecturale)

2.2.2 Traitement des donnnées

Avant de donner à un réseau de neurones quelconque données récupérées pour son entraînement, il est nécessaire de préparer ces données. Le but étant de rendre ces données d'entrées correctes et compréhensibles pour notre intelligence artificielle. Cet préparation s'effectue suivant le type de tâche souhaité. Néanmoins, dans sa généralité, les étapes de préparation de données en TAL restent les mêmes.

Tout d'abord, il faut normaliser (nettoyer) notre 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 [2] Dans ces données se trouvent évidemment des mots, avec des ń caractéristiques ż de la langue [3] tel que les ponctuations, les lettres en capitales, les chiffres, ou bien les caractères spéciaux (comme le dièse que l'on retrouve souvent dans les *tweets*). Seulement, il arrive parfois que certains de ces éléments rajoutent inutilement de la complexité pour une machine, sans que cela apporte plus de sens, ou bien sont non-désirées (voir incorrectes), ainsi on décide de les supprimer (ou les remplacer) pour ne garder que ce qui nous intéresse. [Exemple]

Après avoir nettoyé notre base de données, il faut segmenter notre texte en mots ou sous-mots, autrement dit, il faut tokeniser notre texte. La tokenisation correspond à la segmentation de chaînes de caractères (comme notre texte) en tokens (mots, sous-mots, ponctuations). Une approche naïve est de tokeniser notre texte en mots suivant les espaces [4]. Prenons un exemple ń J'aime les bananes. ż tokenisé cela donne [ń J'aime ż, ń les ż, ń bananes. ż]. Cependant, cette approche pose des problèmes, en commençant par le token ń bananes. ż 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 ń J'aime ż les tokens [ń J ż, ń ' ż, ń aime ż] 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 ń J'aime ż et de le transformer en deux tokens [ń Je ż, ń aime ż]), 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 [5] 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* [6] (Kudo et Richardson, 2018).

2. Dans le cadre de ce mémoire, le but n'est pas de savoir comment on parvient à extraire du texte à partir de ces sources, mais plutôt de savoir comment rendre ce texte compréhensible à une intelligence artificielle choisie.

3. Pour simplifier, nous avons pris l'exemple d'une base de données monolingue, mais il est possible qu'elle comporte plusieurs langues. Dans tout les cas, le procédé sera le même, si ce n'est qu'il faut s'adapter à chaque langue.

4. Vous remarquerez déjà que ce processus ne s'applique pas au langue comme le Japonais, ou le Chinois.

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

6. Cet algorithme est une implémentation des deux premiers.

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 une chaîne de caractères (mots, tokens) en un vecteur, 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*).

Discuter rapidement du one hot encoding, puis du word embedding (statique et contextuel)

Data splitting and batching

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.

Réseaux de neurones récurrents

...

Transformeurs

...

Chapitre 3

Les contributions de l'IA dans la linguistique historique

C'est la partie 'Related Work', elle discute des différents aspects où la linguistique historique s'applique, à travers différents modèles.

3.1 Restoration de documents anciens

3.2 Déchiffrement de langues anciennes

Chapitre 4

Étude du cas de l'application de l'IA pour la reconstruction des proto-formes d'une langue

Ici, on se place dans un cas concret, pour montrer que ce n'est pas que de la théorie. En proposant une expérience.

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).

4.1.2 Dernières solutions neuronales

Solution supervisée + 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 Observation expériementale d'une limite d'applicabilité d'une approche

4.2.1 Méthode

4.2.2 Récupération de la base de données

Il est fort possible que cette partie se regroupe avec la partie suivante, car il n'y aura pas grand chose à dire.

4.2.3 Normaliser les données

4.2.4 Analyse

4.2.5 Critiques

Il reste ici quelques sous parties à détailler.

Chapitre 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.

Chapitre 6

Références

6.1 Bibliographie

Bibliographie

[1] James H. Martin DAN JURAFSKY. « Neural Networks and Neural Language Models ». In : *Speech and Language Processing*. 2023. Chap. 7, p. 1, 7. URL : https://web.stanford.edu/~jurafsky/slp3/7.