MASTER EN SCIENCES INFORMATIQUES

Justice prédictive

La recevabilité d'une requête d'appel d'un jugement

NTABUHASHE CIZIGE, Gerard

Award date:
2022

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2021-2022

Justice prédictive
La recevabilité d'une requête d'appel d'un jugement

Gérard Ntabuhashe

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Marie Jacquet

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

L'objectif de ce travail est de vérifier, par l'élaboration d'un modèle de classification, dans quelle mesure l'intelligence artificielle peut être utilisée pour rendre le processus juridique plus efficace. Se basant sur la jurisprudence existante, le modèle ainsi élaboré devrait avoir la capacité de prédire la recevabilité ou non de l'appel d'un jugement. Dans ce processus de création, le modèle de langage a été créé et a été ajusté à la compréhension de la jurisprudence. Par contre, la création du modèle de classification est apparue impossible. Ce constat a été fait suite à l'impossibilité de constituer ou de trouver un jeu de données permettant d'aboutir à la création du modèle de classification.

Ayant constaté la qualité insuffisante dans la publication de la jurisprudence en Belgique ainsi qu'à un format de publication disparate au sein de l'Union Européenne, ce travail procède à la proposition d'une norme définissant les méta-données à rendre disponibles lors de la publication d'une décision de justice.

Table des matières

Table des figures

Liste des tableaux

Préface

L'informatique est pour moi un art. Ma passion pour cet art est née il y a environ 18 ans. Depuis, celle-ci n'a cessé de croitre au fil du temps. Tout d'abord à cause de toute les possibilités qu'elle offre, mais aussi grâce aux personnes rencontrées tout au long de ces années.

Mont intérêt pour la place de l'informatique au sein de la justice est né lors de mon passage professionnel dans le secteur public. Au début, il était question de l'informatisation de la justice belge, un vaste domaine qu'il serait impossible de couvrir dans un seul travail. Très vite le sujet a évolué vers l'intelligence artificielle, en particulier le traitement du langage naturel et l'utilisation qui pouvait en être faite.

Au fil de mes découvertes, cela a fait naitre pour moi une passion pour cette discipline, eu égard à toutes les possibilités qu'elle offrait. J'espère que la lecture de ce travail fera naître au lecteur un intérêt et pourquoi pas une passion pour ce domaine. Car, après avoir réalisé ce travail, j'ai encore envie d'en découvrir plus, de constituer des jeux de données et de travailler la mise à disposition de ceux-ci pour les autres.

Je tiens à remercier le professeur Jean-Marie Jacquet qui m'a dirigé et accompagné dans la réalisation de ce travail. Sans ses conseils et lumières, il aurait été impossible de le mener à bien. Ces derniers m'ont été d'une grande utilité quant à la direction à prendre dans sa réalisation. Mes plus profonds remerciements à ma famille, mes amis et collègues, qui m'ont encouragé et soutenu. Un merci tout particulier à Virginie Valentin et Steven An qui ont consacré du temps à la relecture de ce travail.

Gerry Ntabuhashe
Belgique
Août 2022

Introduction

L'objectif de ce travail est la recherche d'une solution pouvant être utilisée pour améliorer le fonctionnement de la justice par l'entremise de l'informatique. Proposer une telle solution implique la compréhension à la fois de ce qu'est la justice mais aussi les technologies pouvant être utilisées. Cette compréhension nous permettra de tirer le meilleur parti de la technologie dans cette quête de l'amélioration que nous nous sommes fixés.

Il est avant tout important de constater l'utilisation grandissante de l'informatique ces dernières années. L'outil informatique est devenu central dans tous les domaines. Selon les statistiques 2021 de Eurostat, nous ne pouvons que constater le pourcentage très élevé (92%)[Eur22] de ménages connectés au sein de l'Union Européenne des 27. Il s'agit d'une évolution considérable de l'ordre de 17% comparé à 2012. Celle-ci montre à quel point l'utilisation de l'informatique est devenue courante ces dernières années influençant de nombreux domaines.

Cette évolution de la société n'a cessé de s'accélérer et une maitrise des outils informatiques est même devenue essentielle pour traiter la quantité de données aujourd'hui disponible. Rater le train moderne qu'est la numérisation peut-être préjudiciable pour une entreprise ou une institution au point d'affecter l'efficacité de son fonctionnement mais aussi le service rendu aux clients ou utilisateurs.

En Belgique, que ce soit dans le commerce, la finance, l'enseignement, le transport, tous les domaines ont aujourd'hui besoin de recourir aux dernières technologies disponibles pour s'améliorer et certaines institutions publiques ne sont pas en reste. Nous pouvons prendre en exemple, les finances avec plus de 89% de déclarations de l'impôt des personnes physiques faites en ligne [1] ainsi que les différentes plateformes de e-gouvernement comme les plateformes e-Santé [2] et mypension.be [3].

Cependant, il est à noter qu'un domaine pose toujours question, si bien que depuis des années plusieurs chantiers ayant pour objectif de l'informatiser ont été initiés sans grand succès. En effet, la justice est l'un des rares domaines où la modernisation semble difficile à mettre en place, si nous nous basons sur les différents articles de presse [Jul18 ; Mer21].

Dans le présent travail, il sera question de *justice prédictive*. Non pas *prédictive* dans le sens prophétique du terme, mais dans celui de la détermination d'une probabilité qu'un dossier nécessite une attention particulière ou pas. En particulier le développement d'un outil permettant à un justiciable ou ses représentants de déterminer, suite à une décision de justice, quelles sont les chances de la recevabilité d'un recours.

Pour mener à bien ce travail, nous allons dans un premier temps nous intéresser à la justice au sens large. Nous y aborderons son informatisation et particulièrement la justice en Belgique.

1. Chiffres du SPF Economie [Eco]
2. Accessible via l'adresse `https://www.ehealth.fgov.be/fr`
3. Accessible via l'adresse `https://www.mypension.be/fr`

Ainsi, en nous focalisant sur la Belgique nous définirons en détail où se situe notre travail et les raisons de ce positionnement. Au travers du deuxième chapitre, nous introduirons l'intelligence artificielle en parcourant des différentes approches et domaines d'applications, en particulier les réseau neuronaux.

Les différentes jurisprudences sont écrites en langage naturel, langage qui est couvert dans le troisième chapitre. Dans celui-ci, nous aborderons la représentation des mots, la modélisation de l'espace vectorielle, ses applications et en particulier les modèles de langages ayant recours aux réseaux de neurones.

Notre objectif est la création d'un modèle de classification qui repose sur l'intelligence artificielle pour permettre de déterminer la recevabilité ou non d'une requête d'appel d'une décision de justice. Les difficultés auxquelles nous avons fait face dans le développement de ce modèle seront présentées plus en détail dans le quatrième chapitre consacré à son développement. Nous verrons que la réalisation d'un tel modèle n'est pas possible à ce stade à cause de la qualité des méta-données publiées.

L'absence des spécifications sur les méta-données [4], qui doivent être publiées à minima, nous a conduit à la rédaction du cinquième chapitre consacré à la définition de celles qui devraient accompagner chaque décision juridique mise à disposition des citoyens dans le cadre de l'open data.

Finalement, après une discussion sur les apports, limites et perspectives, nous conclurons ce travail.

4. Donnée permettant de parfaire la définition d'une autre

CHAPITRE 1

La justice

La justice est un élément central et essentiel à toute société démocratique. Dès lors sa définition, ainsi que celle du système judiciaire, est à prendre en compte pour la définition des objectifs de l'intégration d'outils informatiques en son sein.

Dans ce chapitre nous allons aborder le sujet de l'intégration de l'informatique dans la justice en parlant des objectifs qui l'accompagnent. Ce chapitre présente aussi quelques exemples d'utilisation de l'outil informatique au sein de la justice ainsi que la notion de justice prédictive.

Notre objectif étant le cas de la Belgique, nous allons ensuite nous intéresser à la justice en Belgique, en fournissant quelques terminologies qui lui sont spécifiques et en abordant son fonctionnement. En particulier le chemin possible que peut prendre un litige entre son occurrence et le moment où le justiciable aura épuisé tous les recours possibles. Comprendre ce fonctionnement nous permettra de positionner la solution proposée par ce travail en son sein.

1 Définitions

1.1 La justice

Dans le dictionnaire Larousse, la justice est définie comme étant la « *fonction souveraine de l'État consistant à trancher les litiges entre sujets de droit et à définir, sur le fondement des lois de la société, les comportements antisociaux* » [Lar21b].

1.2 L'informatisation

L'informatisation est l'action d'informatiser. Ce dernier mot peut, selon le dictionnaire Larousse [Lar21c], être défini comme :

— « *Doter un service, un organisme de moyens informatiques* ».
— « *Assurer la gestion d'un service, organisme par des moyens informatiques* » .

2 Informatisation de la justice

A présent que ces définitions sont établies, quels objectifs doivent être considérés lors de l'informatisation de la justice en particulier ?

2.1 Objectifs

Chaque domaine étant différent, l'informatisation d'un service va poursuivre des objectifs différents. En effet, cette informatisation va devoir prendre en compte les spécificités des processus métier pour définir les buts à atteindre mais aussi les obligations légales qui y sont liés.

Dans le monde médical, l'informatisation va par exemple avoir pour but de faciliter l'accessibilité aux données des patients tout en s'assurant de la confidentialité et la sécurité de celles-ci. Dans une entreprise, il s'agira d'informatiser les processus grâce à un ERP [1] ou les relations clients grâce à un CRM [2]. Dans la finance il sera question de s'assurer du respect des procédures, mais aussi du bon fonctionnement de celles-ci.

En ce qui concerne la définition des objectifs de l'informatisation de la justice, Stefan [Ste10] propose de se focaliser sur les finalités suivantes :

1. « *Rendre les procédures efficaces* par l'accélération des procédures, la réduction de la durée des procès, la standardisation des procédures et le remplacement de la sauvegarde et gestion papier des données par les dossiers numérisées. »
2. « *Transparence* en facilitant l'accès des citoyens, médias et organismes aux informations judiciaires mais aussi en mettant en place des services électroniques à destinations des citoyens, avocats et professionnels du système judiciaire et une communication respectant un standard avec les différents systèmes informatisés impliqués. »
3. « *Protection des données* par la sécurisation des réseaux locaux et public de sorte à ce que les données ne soient accessibles qu'à ceux ayant les autorisations nécessaires en fonction de leurs compétences. »
4. « *Suppression de la corruption* par une assignation aléatoire des dossiers aux juges et procureurs tout en bloquant les accès non autorisées aux données relatives aux dossiers d'instruction. »
5. « *Gestion efficace des ressources* en informatisant leur gestion. Celle-ci devra s'accompagner d'un plan d'investissement pluriannuel basé sur les performances et analyses prévisionnelles. De plus, il faut s'assurer que le patrimoine est géré de façon approprié par les différentes personnes impliquées dans sa gestion. »

Quel est l'état de l'informatisation de la justice, existe-t-il des exemples de cette informatisation ayant relativement permis d'atteindre ces objectifs ou une partie d'entre elles ?

2.2 Exemples d'utilisation de l'outil informatique

Si nous regardons la définition de l'informatisation, nous pouvons dire que l'informatisation de la justice est assez avancé dans le monde et particulièrement dans la société actuelle où l'informatique joue un rôle de plus en plus important. Presque tous les tribunaux du monde utilisent l'outil informatique à un certain niveau. Nous observons que l'époque de l'utilisation de la machine à écrire pour la rédaction des documents juridiques est révolue.

De plus, avec la pandémie de la COVID 19, nous pouvons observer la généralisation de cette utilisation, comme par exemple la tenue des audiences en vidéo conférence, la transmission des contradictions par courriel, l'utilisation de plateformes électroniques pour la transmission des décisions de justice, etc.

1. Enterprise Resource Planning
2. Customer Relationship Management

Il est à noter que bien avant ce bouleversement, certains pays avaient déjà pris le pas d'aller plus loin dans l'utilisation de l'outil informatique au sein de la justice. Au sein de l'Union Européenne nous avons, depuis 2014, un guide sur l'utilisation de la vidéo conférence dans les procédures juridiques transfrontières [eC14].

Allant bien plus loin que le guide de l'union européenne nous pouvons citer la Chine qui, depuis 2017, a mis en place les tribunaux en ligne [Fan18 ; Age17]. Dans ces cours, accessibles à tout moment, toute la procédure se déroule en ligne. L'objectif est de faciliter l'accès aux procédures judiciaires.

Aux États-Unis, dans certains états, l'intelligence artificielle est utilisée pour déterminer la probabilité de récidive des justiciables [LA16]. Nous discuterons des problèmes liés à une telle utilisation plus tard dans ce travail.

Abordons maintenant la justice prédictive dans son ensemble et quel rôle elle pourrait jouer l'intelligence artificielle dans le monde juridique face à la masse de données disponibles.

2.3 La justice prédictive

Nous parlons de justice prédictive, mais qu'est-ce qu'est la justice prédictive ? Nous allons essayer d'apporter une tentative de réponse dans cette sous-section. Dans le terme *justice prédictive* nous reconnaissons aisément les termes *justice* et *prédire*. La définition du premier terme nous l'avons présenté dans la section 1.1 de ce chapitre. Le deuxième terme quant à lui peut être défini, comme « *Annoncer d'avance ce qui doit arriver, par intuition, raisonnement ou conjecture* »[Lar22].

Cette notion de prédire l'issue d'une affaire juridique n'est pas vraiment neuve. En effet, le concept même de jurisprudence n'est il pas de considérer les précédentes décisions de justice comme pouvant permettre d'anticiper les décisions futures ? Cette notion impliquerait que la connaissance de celles-ci rendrait possible la détermination de l'issue probable d'un procès en amont. De surcroit l'article *« The Path of the Law »*, HOLMES juge à la cour suprême des États-Unis, définit la science du droit comme étant « *la prédiction de l'incidence de la force public par l'intermédiaire des tribunaux* » [Hol97] .

L'objectif de la jurisprudence est de garantir la cohérence et l'égalité faces aux litiges pouvant arriver. En effet, il ne serait pas juste que deux litiges identiques produisent deux réponses différentes. Il faut cependant nuancer cette notion de jurisprudence. Cette nuance est à prendre en compte en fonction du système de droit considéré.

Dans le système anglo-saxon, le juge doit suivre la jurisprudence à moins qu'on ne lui démontre que le cas qui lui est soumis est différent du cas d'espèce se trouvant dans la jurisprudence [Edw16]. Dans le droit continental par contre, c'est totalement différent. Dans ce dernier, la jurisprudence est considérée comme un source de droit au même titre que les articles ou théories juridiques scientifiques. Le juge se basera plutôt sur les textes des lois et la jurisprudence ne sera qu'un argument soumis à l'appréciation [Van18].

Dans le droit continental, système auquel la Belgique appartient, « *pour établir une jurisprudence constante il faut plusieurs décisions similaires* » alors qu'en droit anglo-saxon une seule décision suffit [Bar16].

Dès lors, pour le sujet qui nous concerne, l'accès aux décisions juridiques précédentes va être cruciale pour avoir la capacité de prédire l'issue d'un procès par l'entremise de l'intelligence artificielle. Cette utilisation de l'intelligence artificielle pour la prédiction d'une décision a déjà été explorée par quelques récents travaux de fin d'études dont entre autres *« Artificial Intelligence*

and Predictive Justice » de DANIA[Dan21] ou encore *« La justice prédictive : Application au droit des marques »* de JOHAN[Joh20].

La masse de donnée juridique rend, pour un humain, la connaissance de toute la jurisprudence impossible. Cependant la possibilité de pouvoir prédire avec un certain degré de confiance la probabilité de l'issue d'un procès pourrait bien être possible grâce à l'utilisation de l'intelligence artificielle.

Bien entendu, la justice étant assez large, l'étendu de ce travail à sera limité à un cas précis qui sera défini dans la section suivante.

3 La justice en Belgique

En Belgique, l'article 10 de la constitution garantit l'égalité des citoyens belges devant la loi. L'organisation de la justice a ainsi été pensée de sorte à ce que chaque citoyen puisse avoir la possibilité de se défendre mais aussi accéder à tous les recours possibles en fonction de la gravité du litige. Préalablement à une tentative d'explication du fonctionnement de la justice belge, la présentation de quelques définitions s'impose.

3.1 Définitions

La définition des concepts clés suivants est importante dans la compréhension fonctionnement de la justice en Belgique.

- **Le droit civil** est « *le domaine du droit qui règle les relation de base entre citoyens. Il est contenu pour l'essentiel dans le Code civil ainsi que dans des lois particulières.* » [Bel22]
- **Le droit pénal** « *définit les comportements interdits et fixe les peines en cas d'infraction (amendes, emprisonnement, saisie, etc.). ... Le droit pénal est consigné dans le Code pénal.* » [Bel22]. Selon l'article 14 de la constitution, personne ne peut être puni si aucune loi punissant un acte n'existe au moment des faits.
- **Un litige** est définie comme « *un différend entre deux ou plusieurs personnes, les un contestant aux autres d'être titulaire d'un droit à l'exercice duquel ils prétendent.* » [Ser22]
- **L'infraction** est « *le comportement sanctionné par la loi pénale, qui peut être une action (ex : le vol) ou une ommission (ex : ne pas porter secours à une personne en danger).* » [Ins22].
- **Le crime** « *est l'infraction définie par le Code pénal comme étant puni d'une peine de privation de liberté de plus de cinq années.* » [Ins22]
- **Le délit** est « *une infraction punie par la loi pénale d'une amende de 26 euros au moins (à multiplier par les décimes additionnels) ou d'un emprisonnement de 8 jours à 5 ans ou encore de ces deux peines cumulées.* » [Ins22]
- **Le droit de rôle** est, selon le ministère de finances, « *une taxe à payer pour l'inscription d'une affaire à l'agenda de une cour ou un tribunal* ». Cette taxe varie entre 50 et 650 euros et dépend du tribunal qui traite l'affaire.

3.2 Fonctionnement

Comprendre le fonctionnement de la justice est important pour déterminer à quel niveau la solution d'informatisation peut avoir tout son sens. Dans la figure 1.1, nous reprenons de façon sommaire le chemin pouvant êter parcouru par un litige ou une infraction dans le système,

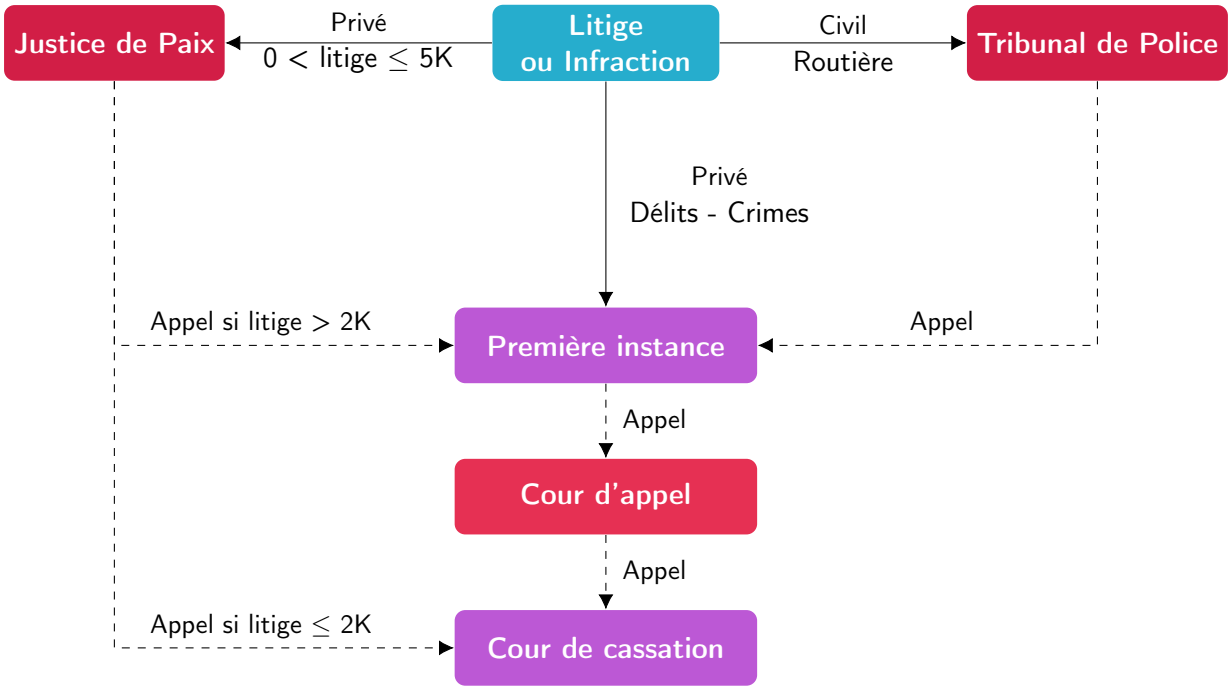


FIGURE 1.1 – Trajet d'un litige dans le système judiciaire belge

particulièrement en ce qui concerne les juridictions judiciaires. Le schéma de cette figure a été réalisée sur la base des explications disponibles sur le site *Questions-Justice.be*[3].

1. *Les litiges* de droit privé dont la valeur est inférieure à 5000 euros sont traité par la *Justice de paix*. Il est possible d'interjeter appel de la décision de justice dans un tribunal de *Première instance* si la valeur du litige est strictement supérieure à 2000 euros. Dans le cas où elle est inférieure, l'interjection de l'appel ne peut avoir lieu qu'au niveau de la *Cour de cassation* on parlera alors d'un pourvoi en cassation.
2. *Les litiges* de droit privé dont la valeur est strictement supérieure à 5000 euros sont traités par le tribunal de première instance.
3. *Les infractions routières* sont traitées par le *Tribunal de police*. Ce dernier traite aussi certains litiges du domaine civil, comme par exemple les demandes de réparations causées par un accident de circulation. Il est possible d'interjeter appel de cette décision dans un tribunal de *Première instance*.
4. *Les délits et crimes* sont directement traités par un tribunal de *Première instance*.

Pour les dossiers qui arrivent dans un tribunal de première instance, si le justiciable n'est pas satisfait du jugement, il a la possibilité de faire appel auprès de la *Cour d'appel*. Une fois passé en appel, la *Cour de cassation* est le dernier recours possible.

La machine judiciaire est très vaste, raison pour laquelle il est important de déterminer à quel endroit notre solution serait intéressante pour améliorer sans entraver le fonctionnement de la justice.

3. Accessible à l'adresse `https://questions-justice.be`

3.3 Positionnement de notre solution

Afin déterminer l'embranchement auquel notre solution serait plus bénéfique au système, nous nous sommes posés les questions suivantes :

1. *A quel point la solution ne va-t-elle pas impliquer un coût pour le justiciable, ou tout du moins lui permettre de réduire la facture qu'il aurait eu à payer pour accéder à la machine judiciaire ?* En effet, il faut noter que l'accès à la justice n'est pas gratuite, rien que le fait d'avoir l'affaire inscrite à l'agenda d'une cour ou d'un tribunal entraine l'obligation de paiement du droit de rôle par une des parties, sans considérer les frais d'avocats. Cette taxe est due même si au final le dossier n'est pas traité par le tribunal. Ceci peut parfois être bloquant pour un justiciable.
2. *Dans quelle mesure la solution va-t-elle permettre de réduire le nombre de dossiers en attente ?* Il nous semble clair que la réduction du nombre de dossier permettrait de s'assurer de l'utilisation efficace des ressources ainsi que la réduction de l'arriéré judiciaire dans certains tribunaux.

En analysant les statistiques des cours et tribunaux pour la période allant de 2012 à 2020 [Col21], nous pouvons remarquer que le nombre d'affaires pendantes au niveau des cours d'appels a diminué passant de 52814 au premier janvier 2012 à 43956 au 1er janvier 2021. Nous illustrons ce constat dans la figure 1.2a. Bien que cette diminution soit de l'ordre de $16,77\%$, elle est à nuancer. En effet, dans la figure 1.2b, nous pouvons observer qu'une diminution a lieu au niveau des cours d'appels des affaires civiles mais les autres cours sont toujours sujettes au même nombre d'affaires pendantes. D'après, les données consultées, le nombre d'affaires traitées est plus ou moins équivalent au nombre de nouvelles affaires, ce qui rend presque impossible d'observer une réduction drastique des affaires non traitées.

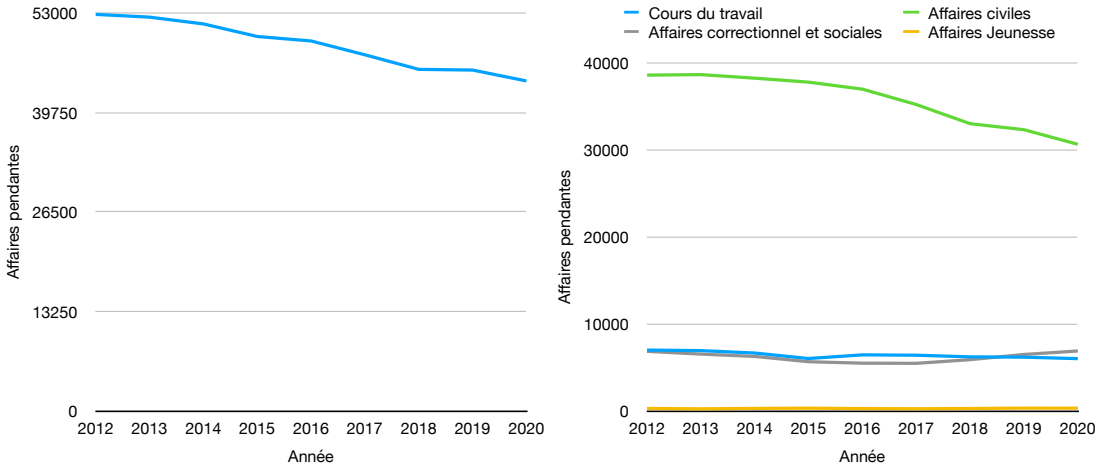
Pour observer une réduction significative il n'y a que deux solutions. Soit réduire le nombre de nouvelles affaires d'une part ou augmenter significativement la capacité de traitement des dossiers d'une autre. Vu l'inefficacité de la dernière solution et les contraintes budgétaires auxquelles fait face le système judiciaire [Inf16], la première solution est celle que notre travail voudrait apporter.

Raison pour laquelle, notre solution est développée pour apporter une réponse bien en amont, entre la décision en première instance et l'introduction de l'appel. Cela permettra au justiciable de ne pas engager des dépenses inutiles en ayant une indication suggestive quant à la recevabilité ou non de son appel bien avant son introduction.

L'objectif premier serait d'éviter l'introduction des requêtes inutiles qui n'auraient que pour conséquence d'engorger les tribunaux déjà bien chargés. En effet, en prenant pour exemple les chiffres 2020 des cours d'appel des affaires civiles, nous noterons un nombre d'affaires pendantes deux fois supérieurs à celles traitées. Un simple calcul, nous permet de conclure que la réduction du nombre de nouvelles affaires de 20% aurait permis celle des affaires pendantes de l'ordre d'environ 4124, ce qui est significatif. De surcroit une réduction de 50% aurait quant à elle induit à environ 7978 affaires pendantes en moins.

3.4 Conclusion

Dans ce chapitre, nous avons parlé de la justice et de son informatisation. La justice prédictive y a été décrite comme n'étant pas vraiment quelque chose de nouveau. En effet la jurisprudence elle même signifie prédire l'issue d'un procès sur la base des précédentes décisions. Nous avons brossé un aperçu du fonctionnement de la justice belge. En analysant les données statistiques



(a) Nombre total d'affaires pendantes (b) Nombre d'affaires pendantes par matière

FIGURE 1.2 – Affaires pendantes - Cours d'appel belges [Col21]

des cours et tribunaux, nous avons pu identifier notre objectif qui est celle de réduire le nombre de requêtes d'appel introduites avec pour finalité la réduction probable du nombre de dossiers pendantes.

Nous allons maintenant, tenter de déterminer les aspects théoriques de la technologie dont nous comptons nous servir pour mener à bien la réalisation de notre travail.

CHAPITRE 2

Intelligence artificielle

La justice prédictive implique l'utilisation d'une machine munie d'intelligence la rendant capable de donner des prédictions. Ce chapitre aborde l'intelligence artificielle en partant de sa définition, ses fondements et des différentes approches à considérer. Ensuite, un regard rapide sera porté sur les différents domaines. Pour finir, nous ferons une analyse des réseaux de neurones, ceux-ci ayant gagné en popularité ces dernières années.

1 Définitions

Avant tout, quelques définitions de l'intelligence artificielle.

Le dictionnaire Larousse définit l'intelligence artificielle comme un « *ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine* » [Lar21a].

D'après le cours « Techniques d'intelligence artificielle » [Jea20, chap. 1], l'intelligence artificielle peut selon M. Minsky être définie comme « *la construction de programmes informatiques qui s'adonnent à des tâches qui sont pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptif, l'organisation de la mémoire et le raisonnement critiqué* ».

2 Fondements

La documentation, dans des domaines très variés, de l'analyse de l'intelligence humaine nous indique des objectifs et des fondements multiples. L'objectif de l'intelligence artificielle étant de simuler l'intelligence humaine, ses fondements se trouveront dès lors dans des origines aussi diverses que :

— *Les mathématiques* pour déterminer ce qui peut-être formalisé, la façon d'arriver à des conclusion valides et comment travailler avec des informations dont la certitude n'est pas acquise.

— *La philosophie* pour essayer de comprendre l'origine des connaissances, la façon dont elles induisent à une action et surtout comprendre l'origine de la pensée.

— *L'économie* pour comprendre pourquoi certaines décisions sont prises tout en étant à contre-courant d'autres décisions dans le but de maximiser le profit non seulement à court terme, mais aussi quand ils ne sont percevables que sur du long terme.
— *La neurosciences* pour ce qui concerne la compréhension du cerveau, comment les neurones communiquent entre eux.
— *La philosophie* pour comprendre comment pensent les êtres humains.
— *L'informatique* : l'ordinateur étant un composant essentiel de l'intelligence artificielle, comment construire un ordinateur disposant d'une grande efficacité ? L'histoire sur la façon dont l'informatique a influencé l'intelligence artificielle est disponible dans [HK19 ; Sap18].
— *La linguistique* pour comprendre le lien qu'il y a entre le comportement verbal et la pensée.

Bien que ces domaines aient eu beaucoup d'influence sur l'évolution de l'intelligence artificielle, il est important de noter que le domaine dans lequel on veut l'appliquer aura une incidence sur sa mise en œuvre. Il faudra prendre en compte les contraintes propres à ces domaines. Par exemple, s'il s'agit de la justice, il faudra considérer les implications qu'auraient une telle utilisation aux principes d'égalité, de motivation des décisions, de compassion, etc.

3 Approches

Toutes ces définitions ont donné lieu à différentes approches. L'ouvrage « *Artificial Intelligence, A modern approach* » [SP10, chap. 1] liste quelques-unes d'entre elles. Bien que celles-ci soient différentes, elles ont fini par se compléter. Elles seront centrées sur l'être humain, en « *ayant recours des observations et hypothèses* », ou alors elles seront « *rationnelles* ».

3.1 Test de Turing

Le test de Turing, comme son nom l'indique est une proposition de test faite par A. Turing dans son le travail *« Computing Machinery and Intelligence »* [Tur50] pour évaluer l'intelligence d'une machine. Ce test implique trois entités distinctes, une machine et deux humains. La machine et un humain devront répondre tous les deux aux questions de l'autre humain. Ce dernier aura pour mission de différencier la machine de l'humain. Si la machine parvient à tromper l'humain ou que celui-ci n'arrive pas à déterminer lequel de ses interlocuteurs est une machine, alors on peut la considérer comme ayant réussi le test.

Une version plus poussée de ce test, « *le test total de Turing* » a été proposé par Harnad dans *« Other bodies, other minds : A machine incarnation of an old philosophical problem »* [Har91]. Il n'est pas seulement question pour la machine de répondre aux questions, mais elle doit aussi pouvoir interagir avec son environnement.

3.2 Agir comme un être humain

Cette approche est la plus complexe à mettre en place car la machine doit alors passer le test de Turing et le test total de Turing. Au travers ces tests, on vérifie la capacité de l'intelligence artificielle à correctement communiquer dans la langue humaine, de sauvegarder ses connaissances, d'interagir sur base de celles-ci, de "*s'adapter aux situations*", de "*percevoir les objets*", de les "*manipuler*" et "*bouger*".

3.3 Penser comme un être humain

Cette approche qui peut consister, selon Stuart et Peter [SP10], en "*introspection*", "*expériences psychologiques*" ou "*imageries cérébrales*", a donné naissance à la science cognitive dont le but est de concilier l'intelligence artificielleet la psychologie.

3.4 Penser de façon rationnelle

L'objectif du penser rationnel est la création d'un programme capable de résoudre tous les types de problèmes. Ici, la difficulté principale consiste à effectuer la conversion d'une connaissance informelle en une connaissance formelle. Une autre, et pas moindre, est le fait que la résolution théorique d'un problème est totalement différente d'une solution dans la pratique.

3.5 Agir de façon rationnelle

L'objectif suivi par cette approche est la création d'agents rationnels capables de faire plus que ce que ferait un simple ordinateur. La représentation de connaissances et le raisonnement peuvent permettre à ces agents de prendre des bonnes décisions. Il est à noter que de façon générale les capacités attendues par le « *Test de Turing* » permettent aussi à l'agent d'agir de façon rationnelle. En effet, cet agent agira de cette façon dans l'environnement qui est le sien ce qui peut compliquer le travail.

4 Domaines

La rencontre de ces approches a permis le développement de plusieurs domaines au sein même de l'intelligence artificielle. C'est le cas par exemple de l'utilisation de l'apprentissage machine dans le traitement du langage naturel, comme nous le verrons plus loin. En effet, tout comme pour l'intelligence humaine, il faut apprendre avant d'avoir les capacité de traiter le langage et cet apprentissage va utiliser les techniques existantes.

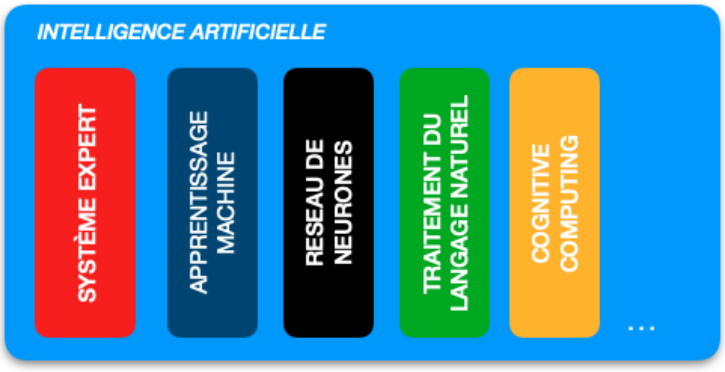


Figure 2.1 – Domaines d'étude de l'Intelligence artificielle[a]

[a] Il s'agit d'une représentation non exhaustive

La figure 2.1 illustre ces domaines de façon non exhaustive. Dans cette section, nous allons en effectuer une exploration rapide.

4.1 Le traitement du langage naturel

Le traitement du langage naturel va permettre aux ordinateurs d'interpréter et de traiter le langage humain et la parole [SP10]. L'objectif du langage naturel est de rendre compréhensible aux machines, un langage aussi complexe et ambigu que le langage humain. Il va avoir recours à la linguistique pour analyser la structure syntaxique, grammaticale et la signification de mots avant d'utiliser des algorithmes pour exécuter différentes tâches.

Nous abordons, de façon détaillée, le traitement du langage naturel ainsi que les algorithmes impliqués dans le chapitre 3.

4.2 Systèmes experts

Les systèmes experts font partie des premiers modèles développés pour l'intelligence artificielle. Feigenbaum et McCorduck [tel que cité dans CYM95] définissent une système expert comme étant « *un logiciel qui utilise les connaissances d'un expert et des procédures d'inférences pour résoudre des problèmes assez complexes nécessitant une grande expertise humaine dans leurs résolutions* ».

Ces systèmes vont simuler la façon dont un être humain disposant d'une connaissance va prendre une décision. La quantité et la qualité des connaissances accumulées vont jouer un rôle important sur l'efficacité d'un tel système.

Un système expert est composé d'une base de connaissances qui est une collection de faits, d'un moteur d'inférence composé de procédures et règles qui vont permettre la déduction ainsi qu'une interface utilisateur qui va être le point de contact avec l'utilisateur du système qui peut ne pas être un expert.

Dans leurs champs d'expertise, ces systèmes seront capables de poser des diagnostics, conseiller, interpréter, expliquer, proposer des solutions alternatives sur un cas, etc. mais ne pourront en aucun cas se substituer à la décision prise par un être humain, donner des solutions fiables si la base de connaissance n'est pas correcte ou encore automatiquement améliorer leurs bases de connaissances.

4.3 Apprentissage machine

L'apprentissage machine va se focaliser sur les techniques permettant à un ordinateur d'apprendre sans avoir à écrire une ligne de code [Sen+18 ; Wik21a]. Cet apprentissage se fera de manière à ce que la machine puisse, grâce à un algorithme, construire sa propre logique sur les données apprises. Il peut se faire en utilisant des algorithmes classés dans deux catégories différentes.

Cet apprentissage peut être supervisé ou non. Les données disponibles seront déterminantes quant au choix du type d'apprentissage à utiliser.

Ainsi on utilisera :

1. **L'apprentissage supervisé** si les données fournies à l'algorithme sont annotées et définissent des variables lui permettant de faire des corrélations. Dans ce type d'apprentissage les données en entrée et sortie de l'algorithme sont prédéfinies.
2. **l'apprentissage non supervisé** si les données disponibles ne sont pas annotées. L'algorithme va déduire les corrélations en analysant les données passées en entrée.

4.4 Réseau de neurones

Cette branche de l'intelligence artificielle va avoir recours à la neurologie avec pour objectif de créer des fonctions qui permettront à la machine d'imiter le fonctionnement du cerveau humain [Wan03].

Le réseau de neurone gagnera en connaissance lors du traitement des différentes données d'entraînement et grâce à l'utilisation des modèles d'apprentissage, il sera possible de fournir des résultats de plus en plus précis.

La composition d'un réseau de neurones et les différents types seront décrits dans la section 5.

4.5 Autres domaines

D'autres domaines, tout aussi intéressants que ceux listés plus haut font partie de l'intelligence artificielle. Ces domaines sont, par exemple, la vision machine [JKS95 ; SQ04 ; SUW18] pour modéliser la vision, la logique floue [KI93 ; Cas+07 ; TE15] pour modéliser l'incertitude, la robotique [HKS17 ; LP17] pour traiter le mouvement des objets par l'ordinateur dans un environnement complexe, etc.

5 Les réseaux neuronaux

Comme illustré dans la figure 2.2, un réseau neuronal est constitué de neurones artificiels et est organisé en couches. Il a une couche d'entrée, peut avoir une ou plusieurs couches cachées, et une couche de sortie. [Abr05]. Un réseau neuronal profond aura plus d'une couche cachée.

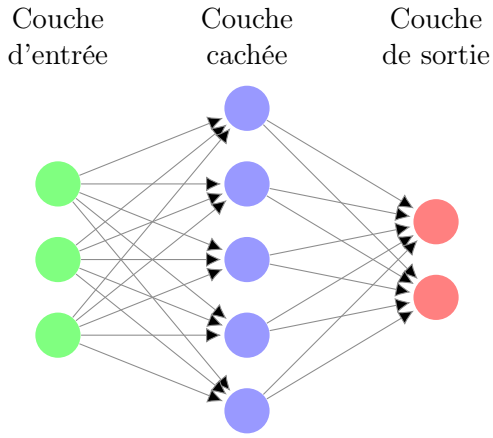


FIGURE 2.2 – Un réseau de neurone

Après avoir reçu les données, les neurones de la couche d'entrée les envoient à ceux de la couche cachée. Ces derniers vont à leur tour les envoyer à la couche de sortie. Mais qu'est ce qu'un neurone artificiel ?

Un neurone artificiel, est composé de synapses (qui sont des connections d'un poids donné), d'une fonction d'activation déterminant la sortie pour une entrée donnée et une sortie. Les sorties des neurones sont différenciées par la façon dont les synapses sont paramétrés. Ce sont ces derniers qui vont produire le signal passé à la fonction d'activation pour obtenir une sortie

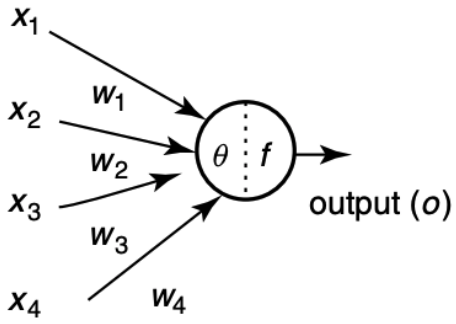


FIGURE 2.3 – Un neurone artificiel (Source : [Abr05])

du neurone. Dans la figure 2.3, $x_1, \ldots, x_4$ correspond aux données passées en entrée, $w_1, \ldots, w_4$ au poids du synapse, O à la sortie. La sortie du neurone sera donnée par la fonction définie en (2.1).

$$O = f\left(\sum_{j=1}^{n} w_j x_j\right) \tag{2.1}$$

Dans le cas où la fonction d'activation est à seuil, ce dernier sera définie par θ de sorte que O aura une définition identique à celle fournie en (2.2).

$$O = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} w_j x_j \geq \theta \\ 0 & \text{autrement} \end{cases} \tag{2.2}$$

Quelles sont ces fonctions d'activation et quelle façon les choisir efficacement ?

5.1 Les fonctions d'activation

L'utilisation des fonctions d'activation est ce qui permet au réseau de neurones d'apprendre et de pouvoir résoudre des problèmes complexes. Le choix d'une fonction d'activation va principalement se baser sur ses caractéristiques telles que sa non linéarité, sa différentiabilité et son étendue.

Les fonctions d'activation populaires sont la fonction linéaire, la fonction échelon unité, la sigmoïde [1], la tangente hyperbolique (**tanh**), la fonction unité linéaire rectifiée (ReLU [2]) ainsi que ses variantes et la $Softmax$. Pour chacune des fonctions précédemment citées, une définition est disponible dans la table 2.1.

Selon les auteurs de l'étude « Activation functions in neural networks »[SS17], choisir la fonction d'activation n'est pas chose facile. Ce choix dépend de la tâche à effectuer et parfois il est nécessaire de tester plusieurs fonctions d'activation avant de trouver celles qui conviennent.

En guise d'exemple, ils suggèrent l'utilisation des sigmoïdes pour les problèmes de classification, la fonction ReLu pour ses meilleures performances par rapport aux autres, leaky ReLu en cas d'existence de neurones morts dans le réseau neural. De plus, ils déconseillent l'utilisation de

1. Aussi appelée *fonction logistique*
2. Abréviation anglaise pour *Rectified Linear Unit*

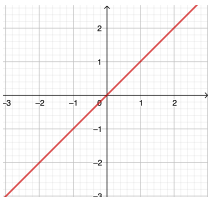
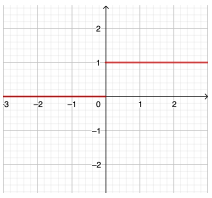
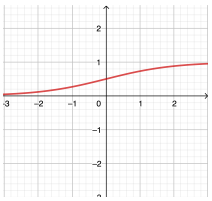
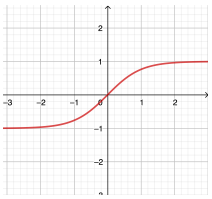
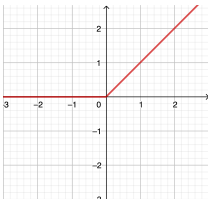
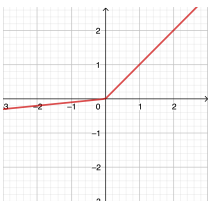
Fonction	Définition	Graphe
Linéaire	$f(x) = x$	
Échelon unité	$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$	
Sigmoïde	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	
Tanh	$f(x) = tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$f(x) = max(0, x)$	
Leaky ReLU	$f(x) = max(\epsilon x, x)$	
SoftMax	$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \ (j = 1, \ldots, K)$	-

TABLE 2.1 – Fonctions d'activation [SS17]

la fonction ReLu dans la couche de sortie ainsi que celle des sigmoïdes et tangentes hyperboliques à cause des problèmes de gradient.

Il est dès lors impossible de créer une liste exhaustive déterminant quelle fonction d'activation choisir ou pas. Pour Glorot, Bordes et Bengio, auteurs de l'étude *« Deep sparse rectifier neural networks »* [GBB11], la fonction ReLu et ses variantes sont biologiquement plus réalistes comparées à d'autres.

L'utilisation des fonctions d'activation n'est pas suffisante pour permettre un apprentissage efficace. Les méthodes basées sur les gradients peuvent être utilisées pour permettre d'optimiser l'apprentissage. Celles-ci permettent un apprentissage plus stable dans le cas où l'étendue de la fonction d'activation est finie et plus efficace dans le cas contraire.

La rétro-propagation du gradient est une des ces méthodes. Mais qu'est-ce que c'est ?

5.2 La rétro-propagation

L'objectif de l'entrainement d'un réseau de neurones est d'atteindre un certain niveau de précision. Dans l'apprentissage supervisé il s'agit d'arriver à minimiser les prédictions erronées. L'erreur de prédiction est donnée par la différence entre les sorties produites et celles désirées. La rétro-propagation du gradient permettra de corriger les poids synaptiques du réseau pour prendre en compte les occurrences des erreurs de prédiction.

Pour expliquer l'intuition derrière l'algorithme de rétro-propagation, un réseau de neurones peut être considéré en posant ce qui suit :

- $0 \leq n \leq N \in \mathbb{N}$ est le nombre de couches dans le réseau neuronal
- $i, j \in \mathbb{N}$ sont respectivement le numéro du neurone de la couche $n-1$ et celui de la couche n
- e est l'échantillon présenté en entrée et a celui attendu en sortie
- $f_j^{(n)}$ est la fonction d'activation du j^{me} neurone de la n^{me} couche
- $w_{ij}^{(n)}$ est le poids synaptique entre le j^{me} neurone de la couche n et le i^{me} neurone de la couche $n-1$
- La sortie du neurone j de la couche n sera notée $O_j^{(n)}$
- $p_j^{(n)} = \sum_i w_{ij}^{(n)} O_i^{(n-1)}$ paramètres tels qu'utilisés dans l'équation neural définie en 2.1

Le calcul de $O_j^{(n)}$ se fait via la formule neurale suivante :

$$O_j^{(n)} = f_j^{(n)}\left(p_j^{(n)}\right)$$

Arrivé en fin de réseau, la sortie r est obtenue. L'erreur ϵ se calcule en comparant ce résultat et celui attendu a et la formule suivante permet le calcul de cette erreur :

$$e_j^{(n)} = f_j^{\prime(n)}(p_j(n))(r_j - a_j)$$

Une fois cette erreur calculée en sortie, il est désormais possible de la propager aux couches précédentes. Cette propagation se fait grâce à la formule qui suit :

$$\epsilon_i^{(n-1)} = f_i^{\prime(n-1)}\left(p_i^{(n-1)}\right)\sum_i w_{ij}^{(n)}\epsilon_i^{(n)}$$

Ensuite, une fois que ces erreurs on été calculées, on effectue la mise à jour des poids synaptiques entre toutes les couches. Cette mise à jour, se fait en utilisant la formule suivante où $\lambda \in [0, 1]$ correspond au taux d'apprentissage du réseau :

$$w_{ij}^{(n)} = w_{ij}^{(n)} - \lambda \epsilon_j^{(n)} O_i^{(n-1)}$$

5.3 Architectures

Différentes architectures de réseaux neuronaux ont jusqu'ici été développées et proposées pour différentes utilisations. Dans l'article *« The neural network zoo »* [LV20], Leijnen et Veen récapitulent de manière assez complète ces différentes propositions. Discuter de certaines architectures par l'entremise d'un aperçu non détaillé semble s'imposer. Pouvant être classées en familles, passons en revue quelques-unes d'entre elles à savoir les réseaux récurrents, les réseaux neuronaux convolutifs ainsi que les *Transformeurs*.

5.3.1 Réseaux neuronaux récurrents

Les réseaux de neurones récurrents [Elm90] sont généralement utilisés dans la traitement du langage naturel tel que la modélisation du langage et le résumé de texte [Li+17], la traduction machine [Cho+14], la reconnaissance vocale [LJL16], la génération de description d'images [KF15], etc.

Les réseaux neuronaux récurrents, contrairement à un réseau neuronal simple, disposent d'un état entre les différents passages et dans le temps. Les informations fournies aux neurones ne sont pas uniquement en provenance de la couche précédente mais aussi celles relatives aux précédents passages. L'ordre d'entrainement de ces réseaux est très important. Fournir en entrée *chat* suivi de *noir* ne donnera pas forcément le même résultat que de fournir *noir* avant *chat*. Cependant ces réseaux sont sensibles à la perte du gradient qui peut causer, avec le temps, une perte d'information.

Pour palier à ce problème de gradient, des réseaux neuronaux dits Long Short Term Memory (LSTM) [HS97] ont vu le jour. Chaque neurone a une cellule de mémoire et trois portes, celle d'entrée, de sortie et celle d'oubli. L'objectif de ces portes est de gérer le flot d'information au niveau du neurone. Celle d'entrée détermine quelle information de la couche précédente sera gardée par la cellule et celle de sortie détermine quelle information concernant l'état de la cellule sera transmise à la couche suivante. Comme parfois oublier est important, la porte d'oubli déterminera quelle quantité de données à oublier à chaque passage. D'autres améliorations de cette architecture pour apporter plus de performances ont été proposées. C'est le cas par exemple de la « *Gated recurrent units (GRU)* »[Chu+14] où les trois portes sont remplacées par une porte de mise à jour ainsi qu'une porte de re-initialisation.

5.3.2 Réseaux neuronaux convolutifs

Généralement utilisés le traitement et la reconnaissance d'images, les réseaux convolutifs ont naturellement été appliqués à d'autres domaines comme le traitement du langage naturel dans la classification de texte [Con+16] par exemple. Leur fonctionnement repose dans la façon dont ils sont appelés. L'idée centrale de leur fonctionnement est la convolution d'une fenêtre déterminée de données.

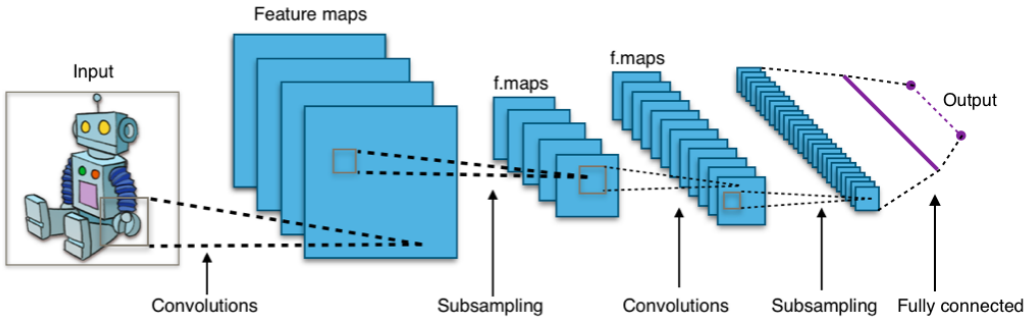


FIGURE 2.4 – Exemple d'architecture d'un réseau neuronal convolutif [Wik21c]

Un réseau neural convolutif [LeC+98] est principalement composé de deux types de couches. *La couche de convolution* dont l'objectif est de trouver les caractéristiques des données et *la couche de pooling* dont l'objectif est de réduire la taille du tableau de caractéristiques.

La convolution est l'étape où ces caractéristiques sont obtenues. Cette opération a donc la responsabilité de détecter celles qui sont les plus importantes. Elle produit en sortie une carte d'activation. Le calcul de cette dernière se fait par l'application d'un détecteur de caractéristiques aux données d'entrée. Le calcul en sortie de la convolution se fait par des opérations de multiplication d'un filtre pouvant être de taille 3×3 et choisi en fonction du problème à traiter.

En imagerie les convolutions sont appliquées au travers de 3 canaux (rouge, vert et bleu). Il est possible d'appliquer un filtre différent à chaque canal pour ensuite assembler les résultats sous la forme d'un seul vecteur. En traitement du langage naturel, les canaux peuvent par exemple être des groupes de mots, un plongement lexical ou un type de mots.

Comme illustré dans la figure 2.4, lors de la conception d'une architecture convolutive, il est tout à fait possible d'avoir plusieurs couches de convolutions ayant entre elles une couche de pooling qui se charge de l'échantillonnage. Cette dernière peut éventuellement être couplée à une couche d'activation généralement appelée *ReLU*.

5.3.3 Transformeurs

Pendant longtemps les réseaux de neurones récurrents, auxquels était ajouté le mécanisme d'attention, étaient la norme en ce qui concerne le traitement du langage naturel. Mais comment expliquer ce mécanisme ?

Imaginons l'espace d'un instant qu'il soit demandé de faire le résumé du concept de « *transhumance* » abordé dans un livre composé milliers de pages. Cet exercice peut se réaliser de deux façons différentes. Soit lire la totalité du livre et en résumer le concept. Ou alors aller dans la table de matière pour trouver le chapitre abordant le sujet pour en faire le résumé. Il est clair que réaliser le résumé en se basant sur la première méthode demandera beaucoup plus de temps que d'utiliser la deuxième. De plus, le résumé découlant de la deuxième méthode sera de meilleure qualité car non pollué par des informations inutiles qui auraient pu être abordées en lisant la totalité du livre.

L'attention fonctionne exactement de cette façon dans les réseaux neuronaux. Pour des longues phrases, au lieu de se focaliser sur tous les mots, le modèle ne se focalisera que sur les mots pertinents présentés en entrée [LPM15].

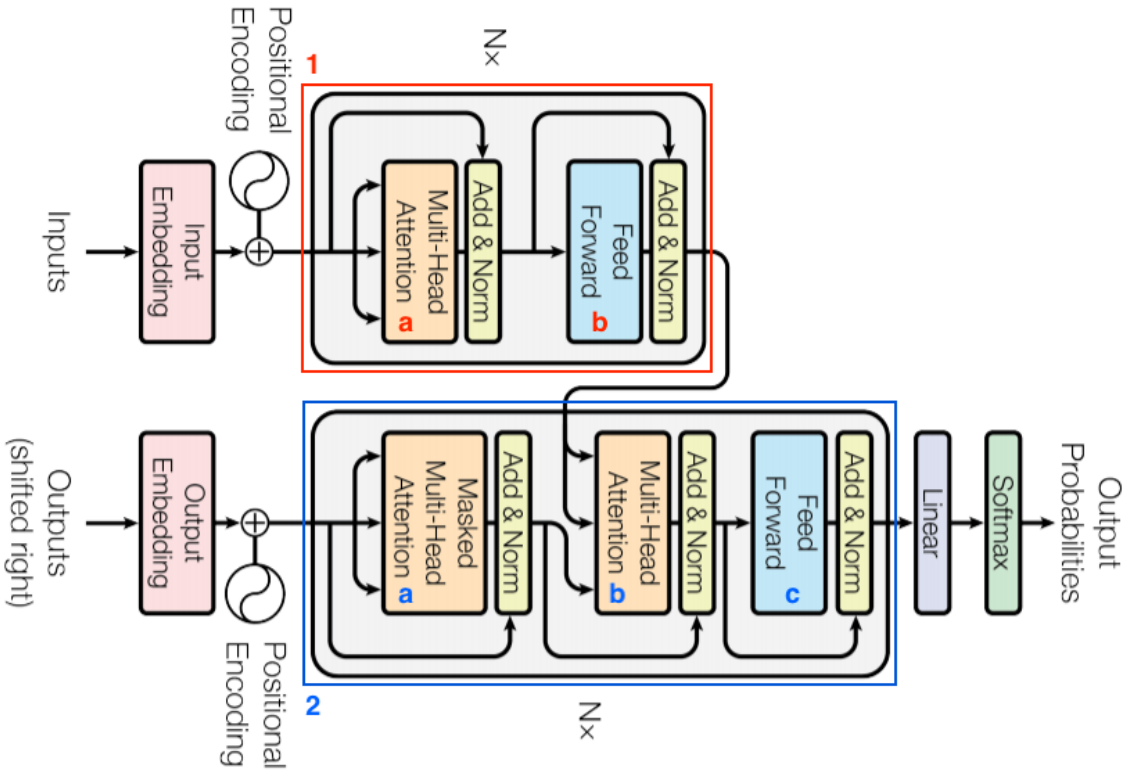


FIGURE 2.5 – Architecture des transformeurs [Vas+17]

La publication par VASWANI et al. dans leur travail *« Attention is all you need »* [Vas+17] bouleverse tout. Dans celui-ci ils réussissent à montrer que les mécanismes d'attention étaient à eux seuls suffisants pour atteindre les performances similaires à celles des réseaux récurrents munis de ce mécanisme. C'était la naissance des transformeurs. Ils arrivent aussi à montrer la possibilité de paralléliser l'apprentissage pour une utilisation efficace du GPU et une accélération de l'entrainement. Un autre avantage de cette architecture est le fait qu'il faut une grande marge pour être affectée par les problèmes de gradient.

La figure 2.5 illustre cette architecture. Mais comment fonctionne-t-elle exactement ? Dans cette illustration, encadré en rouge se trouve le bloc d'encodage (*l'encodeur*) et en bleu le bloc de décodage (*le décodeur*).

1. **L'encodeur** : Généralement les machines ont du mal à comprendre les mots, il est donc nécessaire de commencer par convertir les mots en une structure compréhensible. Il peut s'agir de chiffres, vecteurs ou matrices. La nécessité de cette conversion se situe dans le fait qu'il sera nécessaire de créer un espace vectoriel où sont groupés les mots ayant le même sens ou étant proches les uns des autres. Dans cet espace, une valeur particulière sera assignée à chaque mot en fonction de sa signification. Étant donné que le sens d'un mot peut dépendre du contexte dans lequel il est utilisé, le *positional encoder* permettra d'avoir un vecteur qui donne le contexte du mot en fonction de sa position dans la phrase. Une

fois la conversion effectuée, les vecteurs créés sont envoyés au bloc d'encodage. Ce dernier est composé du bloc « *Multi-Head Attention* » et d'un simple réseau neural à propagation vers l'avant (réseau FF ou FFN).

a) « *Multi-Head Attention* » : Dans ce bloc, s'opère la détermination du niveau de pertinence d'un mot par rapport aux autres présents dans la phrase. Pour chaque mot, plusieurs vecteurs d'attention sont générés permettant d'avoir la relation contextuelle entre les mots au sein de la phrase. Pour chacun des mots, le vecteur d'attention final est trouvé en calculant la moyenne de tous les vecteurs correspondants.

b) « *Réseau Feed-Forward* » : A chaque vecteur d'attention, un simple réseau neural FF est appliqué. L'objectif de celui-ci est la transformation du vecteur d'attention en un format compatible avec la couche suivante (d'encodage ou de décodage selon les cas). Bien que ce réseau n'accepte qu'un vecteur à la fois, leur indépendance rend possible un traitement parallélisé.

2. **Le décodeur** : Dans le cas de la traduction machine, l'entrainement du modèle nécessite la fourniture aussi bien de la langue d'origine que de la langue cible. Les phrases sources sont passées dans l'encodeur et celles de la langue cible dans le décodeur. Tout comme pour l'encodeur, une conversion préalable des mots en vecteur est nécessaire et le résultat ainsi obtenu est transmis au bloc de décodage.

   a) « *Masked Multi-Head Attention* » : Comme dans l'encodeur, les vecteurs précédemment créés arrivent dans ce bloc pour la création des vecteurs d'attention. Ici, ces derniers sont créés pour tous les mots des phrase de la langue cible pour pouvoir déterminer la relation entre les mots des différentes phrases. L'appellation *Masked* vient du fait que, lors de la traduction, un mot de la langue source est donné et traduit en fonction des résultat précédents. Ensuite, il est comparé avec le mot passé au niveau du bloc de décodage. Après comparaison, la matrice de traduction est mise à jours de sorte qu'après plusieurs itérations, les bonnes valeurs seront facilement trouvées.

      Raison pour laquelle, il est nécessaire de masquer les mots suivants dans la langue cible pour que lors de l'apprentissage, le modèle doive d'abord prédire le prochain mot en utilisant les résultats précédents.

   b) « *Multi-Head Attention* » : Les vecteurs produits dans la couche précédente et ceux provenant de l'encodeur arrivent dans ce bloc. Ayant un vecteur pour chaque mot de la langue source et cible, le bloc établi la relation liant entre-eux les mots des deux langues. Pour les deux langues, le bloc produit pour chaque mot un vecteur d'attention représentant la relation avec les autres mots des deux langues.

   c) « *Réseau Feed-Forward* » : Le résultat produit au point précédent, est passé à un réseau FF pour être converti en quelque chose compatible avec la couche suivante.

   Le résultat du décodeur est ensuite passé à une couche dont la fonction d'activation linéaire a pour travail d'étendre la dimension en nombre de mots dans la langue cible après traduction. Son résultat est ensuite passé à une couche munie d'une fonction d'activation *Softmax*. Cette dernière se chargera de transformer l'entrée en une distribution de probabilité, la sortie du transformeur.

6 En résumé

Dans ce chapitre, nous avons pris le temps d'en apprendre un peu plus sur l'intelligence artificielle, les différentes approches ainsi que ses principaux domaines. Nous avons en particulier pris le temps d'en apprendre davantage sur les réseaux de neurones. En effet, ce domaine est important car il entre en jeu dans presque tous les autres domaines de l'intelligence artificielle.

Notre objectif est d'utiliser l'intelligence artificielle dans le champ de la justice. Il nous semble dès lors important d'utiliser les réseaux de neurones pour essayer de s'approcher du raisonnement humain dans le cadre de décisions judiciaires. Ces dernières étant rédigés en langage humain, nous allons dans le chapitre suivant aborder le langage et l'utilisation des réseaux neuronaux dans son traitement.

CHAPITRE 3

Le traitement du langage naturel

Le langage est ce qui permet à l'être humain de communiquer en vue de se faire comprendre par ses semblables. Dans un monde où la machine joue un rôle grandissant, la quantité d'informations disponibles est devenue tellement grande qu'il est quasi impossible d'effectuer manuellement certains traitements.

Le recours à la machine pour traiter, classer, retrouver et proposer, ainsi que le besoin d'automatiser les procédures devient de plus en plus nécessaire dans tous les domaines. C'est dans ce contexte que le traitement du langage naturel trouve tout son intérêt. Avec les bibliothèques en ligne, les réseaux sociaux et toutes les informations aujourd'hui devenues disponibles, l'importance pour la machine de comprendre toutes ces données grandit jour après jour et présente un grand défi.

Ce chapitre aborde le langage naturel et la façon dont, avec ses subtilités, il est compris par la machine. Avec pour but la compréhension des modèles de langages munis du mécanisme de transfert, cette exploration se fera en partant de la représentation des mots en passant par la modélisation de l'espace vectoriel, les étapes et applications du traitement des langages naturels.

1 Le langage naturel

Chomsky définit un langage comme « *un ensemble fini ou infini de phrases qui à leurs tours sont finis en longueur et constituées d'un nombre fini d'éléments* » [Cho57, p. 13].

Comme décrit dans la section 4.1, le traitement du langage naturel a pour objectif de permettre à la machine d'interagir avec l'être humain en utilisant le langage naturel. Cependant, vu la complexité du langage humain, il faudra doter la machine de la capacité de reconnaitre les mots. De plus, elle doit pouvoir reconnaître leurs sens en fonction du contexte. En effet, un texte est généralement composé de paragraphes composés de phrases décomposables en mots formés de caractères, mais ceux-ci n'ont de sens que s'il sont dans un certain ordre mais aussi un contexte spécifique.

Par exemple, le mot "*louer*" va selon le contexte signifier "*faire l'éloge*" ou "*prendre en location*" ou encore le mot "*clarté*" pourra signifier "*lumière, transparence ou blancheur*" en fonction du contexte.

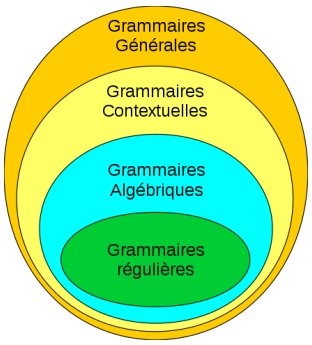


FIGURE 3.1 – Hiérarchie de Chomsky [Wik21b]

Cette importance de prendre les mots dans leur contexte est apparue dès le début de l'étude du domaine. Les approches très simples telles que la traduction "*mot à mot* ont montré leurs limites. Dans [Kum11] un exemple, assez intéressant, de traduction par un tel système[1] est donné. La phrase « *the spirit is willing, but the flesh is weak* » traduite de l'anglais au russe puis du russe à l'anglais par « *the vodka is good but the meat is rotten* ». Le langage naturel étant ambigu à plusieurs niveaux, comment faire pour que la machine puisse le comprendre et se faire comprendre ?

Pour comprendre le langage naturel, la première idée serait d'utiliser un analyseur comme on peut le faire pour les langages de programmation. Cet exercice nécessiterait la connaissance de la grammaire du langage naturel. De plus, vu l'inexistence de spécification du langage naturel, un nombre énorme de possibilités existe.

1.1 La hiérarchie de Chomsky

Dans [Cho56], Chomsky a proposé une hiérarchie des grammaires connue aujourd'hui sous l'appellation hiérarchie de Chomsky et illustrée dans la figure 3.1. Mais quelle place occupe le langage naturel dans celle-ci ?

1.1.1 Le langage naturel n'est pas régulier

[Tel que cité dans NOC11], CHOMSKY avait déjà démontré que le langage naturel n'était pas régulier. Une preuve intuitive est la suivante.

Soient $\mathcal{F}$ le langage définissant la langue française, $a =$ "$Cet\ homme$", $b =$ "$qui\ a\ vu\ l'homme$", $c =$ "$devenir\ fou$", $d =$ "$est\ mort\ de\ folie$". Il va de soi que les phrases illustrées dans la figure 3.2 font partie de la langue française. Avec a, b, c et d, il est possible de définir un langage régulier $\mathcal{L}$ comme dans l'équation (3.1), d'où $\mathcal{F}$ n'est pas régulier.

$$\mathcal{L}(E) = L(ab^*c^*d) \tag{3.1}$$

$$\mathcal{F} \cap \mathcal{L}(ab^*c^*d) = \mathcal{L}(ab^nc^nd) \text{ avec } n \geq 0 \tag{3.2}$$

De par les propriétés des langages réguliers, l'intersection entre deux langages réguliers l'est aussi. Si $\mathcal{F}$ était régulier, le résultat (3.2) devrait l'être, or ce n'est pas le cas.

1. Il s'agissait d'un système de traduction développé par une équipe de l'université de Georgetown

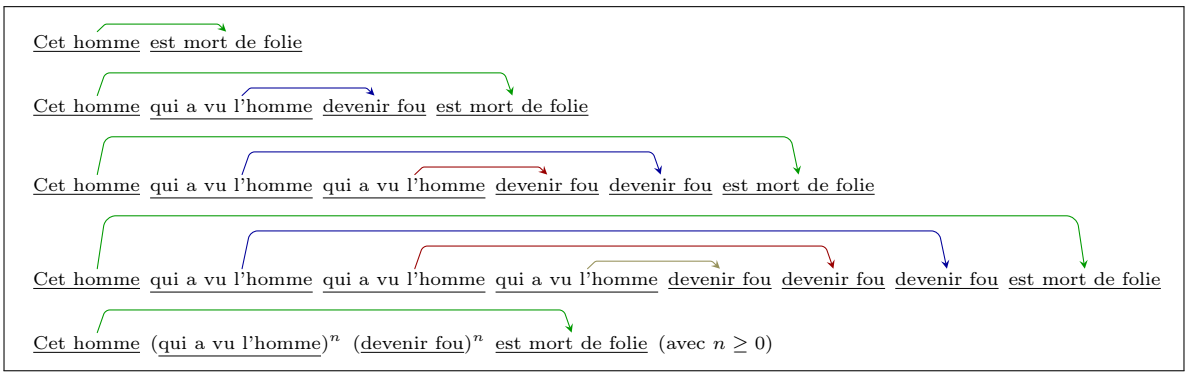


FIGURE 3.2 – Exemple de phrase de la langue française

1.1.2 Le langage naturel est contextuel

Il a été prouvé que certains langages naturels n'étaient pas hors contexte. Cette preuve a été apporté pour le néerlandais [Bre+82] et l'allemand helvétique [Shi85]. En ce qui concerne l'anglais, une preuve qu'il n'était pas hors contexte a été donnée par HIGGINBOTHAM [Hig84].

Dans [DL13], le *disquotational indirect quotation test (IDR)*, *collectivity test* et les *agreements tests* sont proposés pour montrer le caractère contextuel du langage naturel ainsi que l'importance du contexte dans les déclarations. Entrons un peu dans le détail de ces tests dont les deux premiers se basent sur le travail de CAPPELEN et LEPORE [CL06] et le troisième sur [CHH+09 ; Sta05].

La « IDR » Si X peut facilement et sincèrement de façon indirecte par disquotation, rapporter une partie u d'une phrase S par un agent A, ie. *A a dit que S* sans prendre en compte le contexte de l'énoncé d'origine alors S n'est probablement pas sensible au contexte. [DL13, Section 4.1]

La « Collectivity » Test *Soient u et u' des déclarations d'une phrase S par A et B. Si, un rapporteur X peut facilement collecter u et u' dans un seul véritable rapport par disquotation indirecte i.e, avec "A et B ont tous deux dit que S" tout en ignorant et en étant indifférent au contexte d'origine, alors il est peu probable que S soit contextuel.*

Les « Agreements test » *Si dans deux contextes différents deux rapporteurs disent S, mais peuvent être rapportés comme étant d'accord alors S n'est pas sensible au contexte. Par contre si l'un dit S et que l'autre dit le contraire de S, alors S est contextuel seulement si ils ne voulaient pas se contredire* [DL13, Section 4.3].

1.2 Dans la pratique

Cependant, bien que le langage naturel ne soit ni régulier ni hors contexte, l'utilisation des modèles réguliers ou hors contexte est privilégié dans la pratique en NLP [JSW90]. Ce choix est fait à cause de la rapidité et la couverture de tels modèles. Mais comment représenter les mots pour en intégrer le contexte ?

2 La représentation de mots

En traitement du langage naturel les mots sont généralement représentés comme des symboles discrets [FZ20]. Ils sont symbolisés sous forme de vecteur de deux manières différentes [LLS20], la représentation '*One hot*" et la représentation "*distribuée*".

2.1 Représentation one-hot

Une représentation one-hot implique de représenter les mots par des vecteurs où le sens du mot est représenté par 1 et le reste 0 comme dans l'exemple 2.1.

Exemple 2.1 *maison* = $[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$, *logement* = $[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$

De façon plus formelle, en considérant un vocabulaire Σ de taille S, chaque mot du vocabulaire serait représenté de façon indépendante par un vecteur $R^{S\times 1}$ avec un seul 1 et des 0 partout ailleurs.

Tous les mots français seraient alors représentés de la façon suivante :

$$m^{a} = \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix}, m^{\grave{a}} = \begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix}, m^{aa} = \begin{bmatrix}0\\0\\1\\\vdots\\0\end{bmatrix}, \ldots, m^{zythum} = \begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix}$$

Cependant cette représentation est lourde dans le cas du langage naturel où le vocabulaire est bien souvent composé de dizaines de milliers mots [2]. En plus de cette lourdeur, cette représentation n'intègre aucune notion de "ressemblance" ou de "différence". En prenant l'exemple de la figure 2.1, dans cette représentation il sera difficile pour la machine de comprendre qu'un lien existe entre "*maison*" et "*logement*". Comment faire pour intégrer cette notion de similarité dans la représentation des mots ?

2.2 La représentation distribuée

Une solution pour faire face à ce problème d'absence de notion de similarité ou différence est d'encoder ces notions dans les vecteurs et en même temps réduire la taille du vecteur à une valeur beaucoup plus faible. C'est ce que va permettre la représentation distribuée. Il sera question de créer les vecteurs en analysant les mots qui apparaissent fréquemment ensemble, mais avec la particularité que contrairement au "*one-hot*" plutôt que d'avoir des 0 et un 1 toutes les valeurs du vecteur seront différentes de 0. De plus ce vecteur sera plus court mais dense. En effet, si un mot donné apparaît dans un texte, le contexte sera donné par les mots qui sont autours [Fir57], par exemple dans la figure 3.3, le mot logement est utilisé dans des contextes différents, qui peuvent avoir une incidence sur la compréhension du texte.

Cette représentation va être importante car elle va permettre le développement de modèles et méthodes qui vont être utiles lors de l'apprentissage de la machine. Les méthodes basées sur la *Décomposition en valeurs singulières (SVD)* [3] [BL12], les modèles *BoW* [4] [SM83], *N-Gram*

2. Le français en compte plus de 50.000 [fra].
3. *Singular Value Decomposition*
4. *Bag of words ou Sacs de mots*

*Le **logement** social est destinée aux personnes isolées*
*il propose un **logement** à louer mais aussi des bureaux*
*la lutte contre l'insalubrité d'un **logement** pose d'innombrables questions*

FIGURE 3.3 – Le mot logement utilisé dans des contextes différent

{'Elle' : 2, 'aime' : 3, 'cet' : 2, 'homme' : 2, 'mais' : 1, 'déteste' : 1,
'son' : 2, 'enfant' : 1, 'qu'il' : 1, 'père.' : 1, 'vraiment' : 1} (3.3)

[2, 3, 2, 2, 1, 1, 2, 1, 1, 1, 1] (3.4)

FIGURE 3.4 – Exemple de création du vecteur par le modèle sac de mots

[Bro+92 ; VVV10], *TF-IDF* [Jon72 ; SM83], les modèles [5] proposés par MIKOLOV et al. dans le framework word2vec[Mik+13] sont quelques exemples pouvant être données.

3 Modélisation de l'espace vectoriel

Une brève analyse de quelques modèles permettant de constituer l'espace vectoriel des mots s'impose pour en déterminer l'utilisation et le fonctionnement.

3.1 Le modèle BoW ou Sac de mots

Le modèle du sac des mots va représenter sous forme de vecteurs la présence de mots dans un texte donné. Ce vecteur sera composé du poids de chaque mot en fonction du nombre de fois que le mot apparaît dans le texte. Dans ce vecteur créé, l'ordre dans lequel les mots apparaissent n'a pas d'importance. Ce modèle va, généralement, être utilisé pour déterminer la fréquence d'apparition de mots dans le document. L'espace vectoriel sera composé des vecteurs représentant tous les documents du corpus.

La figure 3.4 illustre la création du vecteur pour le texte « *Elle aime cet homme, elle déteste tout ce qu'il aime* ». Dans celle-ci, le comptage des différents mots présents dans le texte est illustré en (3.3) et le vecteur représentant ce document avec la pondération des différents mots en (3.4).

3.2 Les Modèles N-Grams

Il s'agit des modèles où pour avoir le contexte d'un mot, on considère les mots ou les caractères de longueur N qu'on peut trouver dans le texte passé au modèle. Les séquences qui vont être créés par ces modèles vont dès lors être composées de $2, 3 \ldots$ mots en fonction de la valeur de N. Le tableau 3.1 montre la décomposition de la phrase "*Tout flatteur vit aux dépends de celui qui l'écoute*" lors de l'établissement de la matrice du document pour $N = 2,\ 3$ et 4.

5. *Continious bag of words(CBoW)* et *Skip-Gram*

N	Tokens
2	['tout flatteur', 'flatteur vit', 'vit aux', 'aux dépends', 'dépends de', 'de celui', 'celui qui', 'qui l'écoute']
3	['tout flatteur vit', 'flatteur vit aux', 'vit aux dépends', 'aux dépends de', 'dépends de celui', 'de celui qui', 'celui qui l'écoute']
4	['tout flatteur vit aux', 'flatteur vit aux dépends', 'vit aux dépends de', 'aux dépends de celui', 'dépends de celui qui', 'de celui qui l'écoute']

TABLE 3.1 – Exemple d'application du modèle N-grams avec $N = 2, 3$ ou 4

L'idée principale de ces modèles est d'avoir l'information sur la probabilité qu'un mot soit suivi par un autre. On aura un contexte plus précis avec une valeur de N plus élevée.

3.3 Le Modèle TF-IDF

Le modèle TF-IDF[6], utilisé pour le calcul du score de pertinence d'un document dans les moteurs de recherche [NBK18 ; Jal+21] ou pour la génération automatique de résumé de texte [CAS16], calcule la fréquence à laquelle un mot apparaît dans un document par rapport à la fréquence d'apparition dans les autres documents. Le TF[7] correspond à la fréquence d'apparition du terme dans le document et le IDF[8] correspond au nombre de fois que le terme apparait dans les documents du corpus.

Telle que représentées dans le tableau 3.2, plusieurs variantes existent pour déterminer la fréquence du terme mais aussi les différentes façons de calculer la fréquence inverse d'apparition d'un terme dans les autres documents comme repris dans le tableau 3.3. Dans ces deux tableaux D est le corpus, d est un document donné, N est le nombre de documents, t est le terme et $n_t = |\{d \in D : t \in d\}|$ le nombre de documents où le terme t apparaît.

Schéma de pondération	Formule du $TF(t, d)$
Binaire	$0, 1$
Fréquence brute[9]	$f_{t,d}$
Fréquence du terme	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
Normalisation logarithmique	$\log(1 + f_{t,d})$
Normalisation "0.5" par le max	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
Normalisation par le max	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

TABLE 3.2 – Variantes de calcul de la fréquence du terme (TF) [Wik21e]

6. *Term Frequency - Inverse document frenquency*
7. Term frequency
8. Inverse document frequency
9. Occurrences du terme dans le document

Schema de pondération	**Formule** $IDF(t, D)$
Unaire	1
IDF	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
IDF smooth	$\log\left(\frac{N}{1+n_t}\right) + 1$
IDF max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic IDF	$\log \frac{N-n_t}{n_t}$

TABLE 3.3 – Variantes de calcul de la fréquence inverse du terme (IDF) [Wik21e]

Document d_1	
Terme	**Occurrence**
aime	3
elle	2
déteste	1
fils	1
père	1
homme	2

Document d_2	
Terme	**Occurrence**
aime	1
ce	1
père	1
détester	1
fils	1
son	1

Document d_3	
Terme	**Occurrence**
qui	1
le	1
père	1
déteste	1
fils	1
son	1

TABLE 3.4 – Représentation de d_1, d_2 et d_3 pour le calcul du TFIDF

La valeur finale du TFIDF sera donnée par le produit de TF et de IDF sous la formule suivante :

$$TFIDF(t, d, D) = TF(t, d)\ .\ IDF(t, D)$$

Soient d_1, d_2 et d_3 trois documents dont le tableau 3.4 représente les termes et leurs occurrences dans ceux-ci ; et le corpus $D = \{d_1, d_2, d_3\}$

Pour cet exemple, *la fréquence du terme* sera utilisée en tant que schéma de pondération pour le TF. Pour l'IDF, le schéma de pondération *IDF* sera utilisé. Le calcul du $TFIDF$ pour le terme « *aime* » se fait comme suit.

1. Calcul du TF

$$TF(''aime'', d_1) = \frac{3}{3+2+1+1+1+2} = \frac{3}{10} = 0.3$$

$$TF(''aime'', d_2) = \frac{1}{1+1+1+1+1+1} = \frac{1}{6} \approx 0.17$$

$$TF(''aime'', d_3) = \frac{0}{1+1+1+1+1+1} = 0$$

2. Le corpus étant composé de 3 documents et le terme *aime* apparaissant dans 2 d'entre eux, le calcul de l'IDF est le suivant.

$$IDF(''aime'', D) = \log\left(\frac{3}{2}\right) \approx 0.18$$

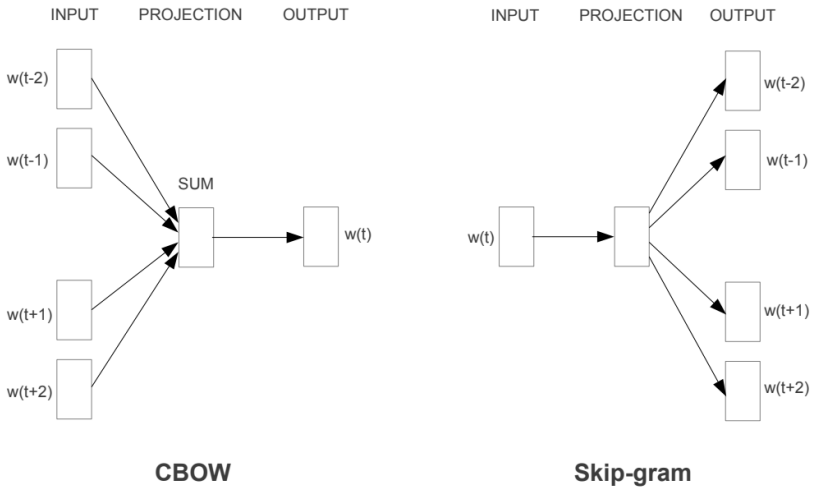


FIGURE 3.5 – Représentation graphique des modèles CBOWet Skip-gram [MLS13]

3. Caldul du $TFIDF$

$$TFIDF(''aime'', d_1, D) = 0.3 * 0.18 = 0.054$$

$$TFIDF(''aime'', d_2, D) = 0.17 * 0.18 = 0.0306$$

$$TFIDF(''aime'', d_3, D) = 0$$

3.4 Modèles du Word2Vec

Deux modèles, utilisés dans l'apprentissage non supervisé et introduits par MIKOLOV et al. ; MIKOLOV, LE et SUTSKEVER, font partie des modèles qui ont fini par permettre les techniques de prédiction en NLP. Ces modèles, très utilisés aujourd'hui, sont le Skip-gram et le Continious bag of words (CBOW). Les techniques utilisées font recours aux réseaux de neurones et intègrent le contexte. Ils sont utilisés en combinaison avec un algorithme FastText [Jou+16]. Ce dernier utilise l'information au niveau du caractère pour générer la représentation d'un mot. Dans cet algorithme, le mot est considéré comme un sac de caractères n-grams en combinaison avec le mot lui-même.

Dans cette section, V sera considéré comme le vocabulaire, $|V|$ sa taille et E celle des vecteurs appartenant à l'espace vectoriel représentant les mots de V.

3.4.1 Skip-gram

Le but du modèle Skip-gram est de pouvoir, considérant un mot donné, déterminer les mots voisins. Cette détermination des voisins se basera sur une fenêtre X donné, pour un mot, déterminer les X mots qui précèdent et qui le suivent. Par exemple étant donnée la phrase « *Je vais au bureau tous les jours pour boucler ce projet* », si le mot cible est *bureau* et la taille de la fenêtre vaut deux, alors les mots voisins seront $\{"vais", "au", "tous", "les"\}$. Et les échantillons d'entrainements pour ce mot seront les paires $("bureau", "vais")$, $("bureau", "au")$, $("bureau", "tous")$ et $("bureau", "les")$. La figure 3.6, où les mots cibles sont en bleu et les

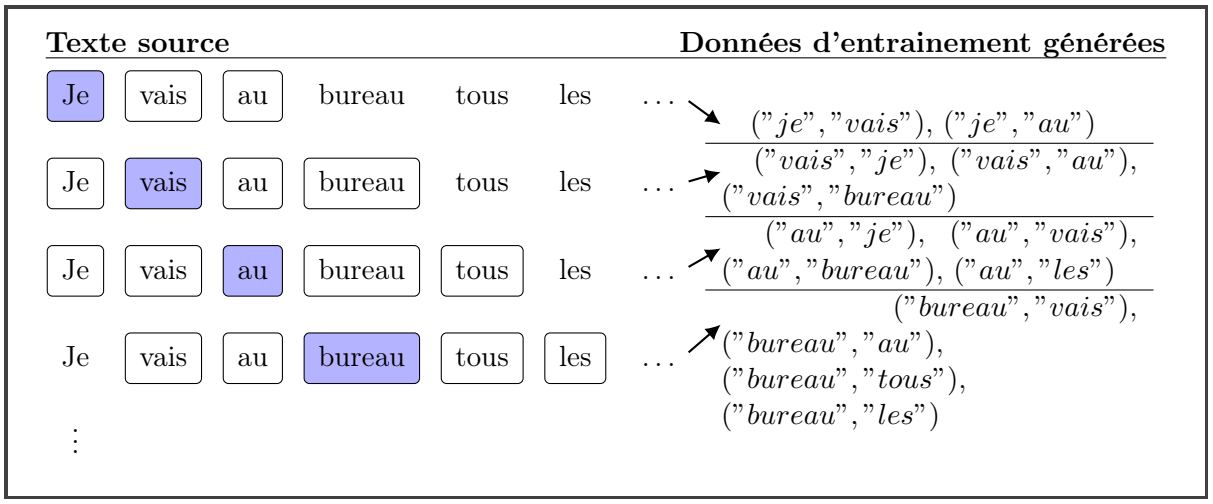


FIGURE 3.6 – Exemple d'entrainement du modèle Skip-gram

voisins sont encadrés, illustre quelques échantillons d'entrainement qui peuvent être tirée de cette phrase.

Les statistiques seront apprises par le réseau de neurones en fonction du nombre de fois qu'une paire de mots est rencontrée, de telle sorte que les mots qui apparaissent souvent ensemble vont avoir une plus grande probabilité de se retrouvent côte à côte.

Soient $w_{c,j}$ le j^{eme} mot prédit dans la position du contexte c, $w_{O,c}$ le mot présent à cette position c, w_I le mot d'entrée, $u_{c,j}$ la j^{eme} value dans le vecteur U lors de la prédiction pour le mot dans la position c du contexte. La *fonction de probabilité softmax*[10] [Wik21d] est définie de la façon suivante :

$$p(w_{c,j} = w_{O,c}|w_I) = \frac{e^{u_{c,j}}}{\sum_{j'=1}^{|V|} e^{u_{j'}}}$$

L'architecture neurale de ce modèle est représentée dans la figure 3.5. Dans cette dernière, $w(t)$ est le mot ciblé. Elle contient une couche cachée qui n'a recours à aucune fonction d'activation. Le vecteur obtenu au niveau de la couche cachée est passé à celle de sortie qui, à son tour, va calculer le produit scalaire entre ce vecteur qui lui a été passé et la matrice des poids. Ensuite, la fonction *softmax* est appliquée pour calculer la probabilité que les mots soient dans le contexte de $w(t)$ à une position donnée.

Ce modèle a l'avantage de requérir peu de mémoire mais le temps nécessaire à son entrainement est très élevé.

3.4.2 CBOW

Le modèle CBOW est utilisé pour déterminer un mot donné en partant des mots qui l'entourent. L'intuition derrière le modèle est présentée dans la figure 3.7. Dans cette dernière, les mots du contexte sont présentés avec un arrière plan bleu et le mot cible rouge. L'architecture neurale de ce modèle est représenté dans la figure 3.5.

10. Fonction exponentielle normalisée

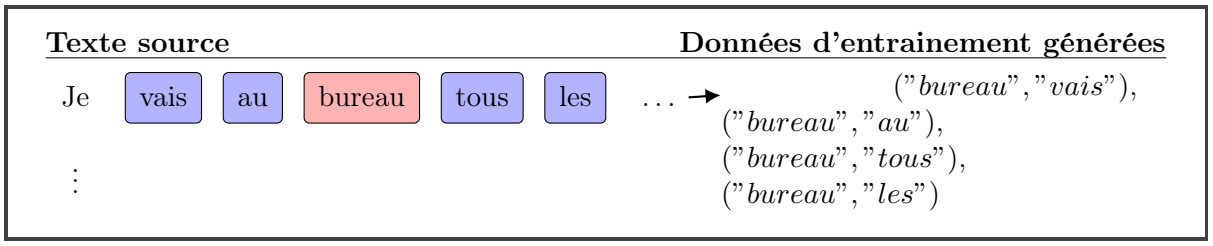


FIGURE 3.7 – Exemple d'entrainement du modèle CBOW

L'apprentissage se fait dans le sens inverse du Skip-gram. En effet, au lieu d'avoir un mot cible en entrée, ce sont les mots du contexte qui sont passés en entrées et transmis à la couche cachée. Les vecteurs obtenus vont être passés à la couche de sortie, qui va en calculer le produit scalaire avec la matrice des poids. Pour finir, comme dans le cas du Skip-gram, la fonction *softmax* sera utilisée pour calculer les probabilités.

Comparé à Skip-gram, ce modèle présente l'avantage d'être plus adapté aux grandes quantités de données et d'être plus rapide lors de l'apprentissage [Mik+13].

3.4.3 L'algorithme FastText

La façon dont cet algorithme [Jou+16] représente les mots permet la reconnaissance des mots rares ou pas encore rencontrés dans le vocabulaire. Le modèle va pouvoir reconnaître les préfixes et suffixes beaucoup plus facilement.

Cet algorithme va par exemple, si $n = 3$ représenter le mot *terme* sous forme n-gram comme étant $< te, ter, erm, rme >$.

3.5 Le modèle Glove

Glove [11] est un modèle de représentation de l'espace vectoriel basé sur un algorithme d'apprentissage non supervisé. La différence par rapport au modèle *word2vec*, présenté dans la section 3.4 est qu'il ne se base pas uniquement au contexte local des mots [PSM14]. Pour trouver le vecteur représentant un mot, aux statistiques locales sont ajoutées les statistiques globales du mot au sein du corpus.

4 Étapes en traitement du langage naturel

Plusieurs étapes sont nécessaires lors de la création d'un modèle en traitement du langage naturel. Comme représenté dans la figure 3.8, en fonction des objectifs suivis, il sera important de suivre quelques étapes lors de toute tâche, et plus particulièrement avant la création de tout modèle et d'entreprendre l'apprentissage. Ces étapes sont dans tous les cas la *normalisation* et la *subdivision en symboles* [12]. De plus, en fonction des objectifs, il faudra aussi effectuer la suppression des mots vides [13] et la lemmatisation.

Tous les exemples de cette section seront donnés en se basant sur le texte de la figure 3.9.

11. Acronyme pour « *Global Vectors* » [PSM14]
12. Aussi connu sous le nom de *tokenisation*
13. Communément connu sous l'appellation « *Stopwords* » en anglais

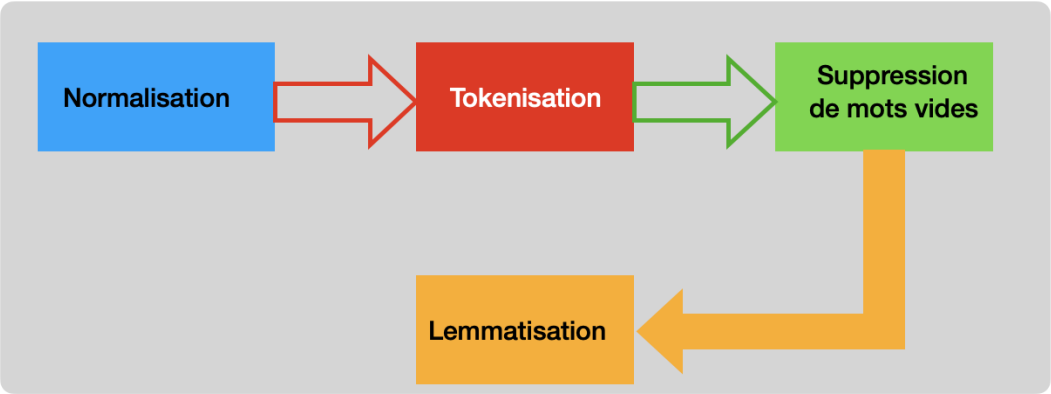


FIGURE 3.8 – Traitement du langage naturel : étapes préalables

Version originale

Le certificat ne se substitue pas aux documents internes utilisés à des fins similaires dans les États membres. Toutefois, dès lors qu'il est délivré en vue d'être utilisé dans un autre État membre, le certificat produit également les effets énumérés à l'article 69 dans l'État membre dont les autorités l'ont délivré en vertu du présent chapitre.

Version normalisé

le certificat ne se substitue pas aux documents internes utilisés à des fins similaires dans les états membres . toutefois, dès lors qu'il est délivré en vue d'être utilisé dans un autre état membre , le certificat produit également les effets énumérés à l'article 69 dans l'état membre dont les autorités l'ont délivré en vertu du présent chapitre .

FIGURE 3.9 – Texte utilisé pour illustrer les différentes étapes du traitement du langage naturel [Cou21]

4.1 La normalisation

Une première étape est d'effectuer la normalisation du texte reçu en entrée pour le dépouiller de tout ce qui n'est pas essentiel, comme le style. Pour un humain les mots *Bonjour*, *bonjour* et *Bonjour !* ont la même signification mais pas forcément pour une machine. A ce stade il faudra effectuer les manipulations nécessaires pour qu'après la normalisation cette distinction ne soit plus possible même pour une machine. Il faudra alors transformer tout le texte en minuscule et le remplacer ou ajouter un espace avant certaines ponctuations (exemple : le `" ."`, le `";"`, le `"?"`, etc.).

A cette étape, peut être fait aussi bien la détection que correction des fautes d'orthographes ainsi que des erreurs de ponctuation. HASSAN et MENEZES ont abordé le sujet dans leur travail *« Social text normalization using contextual graph random walks »* [HM13].

4.2 La tokenisation

La tokenisation est considéré comme étant l'étape initiale dans le traitement du langage naturel [WK92]. A ce stade, il va être question de subdiviser le texte en phrases et celles-ci en mots ou symboles. Un symbole est la plus petite unité qui sera facile pour la machine de comprendre et traiter. Ainsi, les symboles du texte donné en exemple sont « *['le', 'certificat', 'ne', 'se', 'substitue', 'pas', 'aux', 'documents', 'internes', 'utilisés',..., 'vertu', 'du', 'présent', 'chapitre', '.']* ».

La complexité de cette étape variera en fonction des besoins de l'application et de la complexité de la langue à traiter [WK92 ; WF94 ; Par+20]. La tokenisation du français sera par exemple plus complexe que celle du chinois.

4.3 La suppression des mots vides

Les mots vides sont considérés comme du bruit dans certaines applications du traitement du langage naturel. Pour celles-ci, il faut retirer ces mots du texte de sorte à ne plus les avoir dans la liste des symboles présents dans le texte. Cela permettra de réduire la consommation en mémoire et en temps de calcul. Cette suppression se fera généralement pour les applications de classification [Mén+05 ; DS10]. En français, `le`, `la`, `et`, ... sont quelques exemples de mots vides. En effet, ils n'apportent aucune distinction entre les textes.

Ainsi, de l'exemple fourni dans la figure 3.9, après le suppression des mots vides il ne restera que le texte suivant « *certificat substitue documents internes utilisés fins similaires états membres. toutefois , dès lors délivré vue être utilisé autre état membre , certificat produit également effets énumérés article 69 état membre dont autorités délivré vertu présent chapitre .* »

4.4 La lemmatisation

La lemmatisation – à ne pas confondre avec la racinisation qui consiste au retrait de préfixe ou suffixe d'un mot pour en garder la racine – est la procédure permettant de convertir un mot dans sa forme basique en fonction du contexte. Par exemple, la lemmatisation des mots ou groupes des mots *transformes*, *transformerons* et *transformions* est le mot *transformer*. Après la lemmatisation du texte de la figure 3.9, le résultat sera « *certificat substituer document interne utiliser fin similaire état membre . toutefois , dès lors être délivrer vue être utiliser autre état membre , certificat produire également effet énumérer article 69 état membre dont autorité avoir délivrer vertu présent chapitre .* ».

Cette étape trouve tout son sens dans les applications de classification, de la création mots clés pour la recherche, etc.

5 Applications

Il existe plusieurs tâches en traitement du langage naturel. Ceux-ci sont, de façon non exhaustive, la classification de texte, la reconnaissance d'entités (NER), l'étiquetage morpho-syntaxique [14], la génération de texte et la traduction machine.

14. Appelé Part-of-Speech Tagging (POS tagging) en anglais, correspond au fait d'« *Associer aux mots leurs informations grammaticales, tel que le genre, la partie du discours, etc.* »[Wik20]

FIGURE 3.10 – Reconnaissance d'entité : Exemple [Par19]

La popularité des réseaux de neurones a permis et facilité des avancements significatifs dans plusieurs domaines et le traitement du langage naturel ne fait pas exception. Cette section va aborder l'utilisation de ceux-ci dans différentes tâches du traitement du langage naturel.

5.1 Reconnaissance d'entité

Introduit lors de la « Message Understanding Conference 6 (MUC-6) »[GS96], la NER est une problématique de l'extraction d'information, telle qu'une personne, une entreprise, etc., qui peut être appliquée tant aux données structurées que non.

Un exemple de reconnaissance d'entités est présenté dans la figure 3.10, où *Barack Obama* est reconnu comme entité de type *PERSON* et *Facebook* de type *ORG*.

Dans le travail *« Neural Architectures for Named Entity Recognition »* [Lam+16], les auteurs présentent deux modèles pour la reconnaissance d'entités. Ceux-ci se basent sur une représentation des mots sous forme de caractères dont l'apprentissage a été fait aussi bien de façon supervisé que non. Selon les auteurs, ces modèles ont été testés sur plusieurs langues et ont montré de bonnes performances.

5.2 Classification de texte

La classification de texte est importante dans plusieurs applications dont l'analyse de sentiment, le filtre l'information, les moteurs de recherches. Les réseaux de neurones sont très utilisés pour ces tâches en traitement du langage naturel et permettent des très bons résultats [Dev+18 ; Lan+19 ; Rad+18 ; Bro+20 ; WK21].

Dans le travail *« Convolutional Neural Networks for Sentence Classification »* [Kim14], l'auteur présente une série d'expérience se basant sur les réseaux neuronaux convolutifs dont l'objectif est la détection des sentiments positifs ou négatifs. Bien que ce travail se limite à une prédiction binaire, il ne s'agit pas d'une limitation des réseaux de neurones. En effet, « *Stanford Sentiment Treebank* » [15] permet de prédire plusieurs sentiments *(très positif, positif, neutre, négative, très négative)* [Soc+13].

Le résultat obtenu par le travail de KIM a permis de montrer qu'après quelques adaptations, le modèle proposé pouvait être utilisé comme extracteur universel de fonctionnalité et donc être utilisé pour différentes tâches de classification.

Dans un autre travail, *« Text understanding from scratch »* [ZL16], les auteurs affirment qu'il est possible d'atteindre d'excellentes performances en utilisant des réseaux neuronaux convolutifs pour la classification de texte sans connaître les mots ni la structure syntaxique ou sémantique du langage. Cette preuve, ils l'apportent au travers de plusieurs expérimentations et obtiennent des précisions supérieures à 95%. Cependant, atteindre d'aussi bons résultats nécessite une très grande quantité de données.

15. `https://nlp.stanford.edu/sentiment/treebank.html`

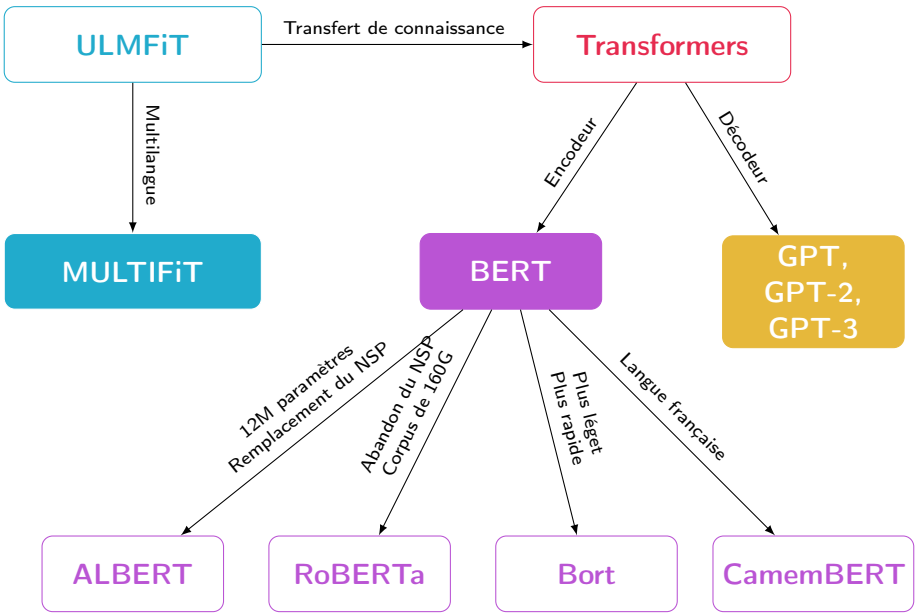


FIGURE 3.11 – Modèles de langage : Architectures, catégorisation et influences

5.3 Génération de texte

La génération de texte est la partie du traitement du langage naturel qui traite de la production d'un texte compréhensible et cohérent. Elle peut être utilisée de différentes façons telles que l'écriture des rapports, les résumés des données médicaux, la rédaction de romans ou encore l'écriture de blagues.

TARASOV décrit un modèle de réseau de neurones récurrents capable de générer les phrases de romans et des résumés de documents [Tar15]. Ce modèle permet aux utilisateurs de contrôler la signification des phrases générées.

5.4 Traduction machine

Malgré les limitations de cette discipline, la traduction machine est utilisée un peu partout dans le monde. Plusieurs modèles ont été testés, dont entre autre l'utilisation des réseau de neurones. Une étude, celle de KRZYSZTOF et KRZYSZTOF, a pour objectif de vérifier les effets de différentes méthodes d'entrainement sur le traduction Polonais-Anglais pour les données médicales. Dans cette étude, l'entrainement du modèle se fait en utilisant les données de l'« *Agence européenne des médicaments (EMA)* » [16].

Grâce à ce travail, il a été montré qu'il n'est pas nécessaire d'avoir beaucoup de données à disposition pour entrainer et maintenir ce réseau de neurones [KK15].

6 Modèles de langage

Un modèle de langage est, en traitement du langage naturel, une représentation de la distribution statistique des mots grâce à laquelle il est possible de prédire le mot suivant un autre

16. Accessibles ici `https://opus.nlpl.eu/EMEA.php`

FIGURE 3.12 – ULMFiT : Étapes dans l'entrainement

mot ou une séquence de mots. Plusieurs modèles de langage ont été proposés. Tous basés sur les réseaux de neurones, ils sont un complément à la section 3. La figure 3.11 illustre les modèles abordés dans cette section ainsi que les influences qu'ils ont eu les uns sur les autres.

6.1 ULMFiT

Quelles seraient les implications si à chaque fois que le cerveau humain, pour apprendre un nouveau concept ou une nouvelle discipline, devait aussi recommencer l'apprentissage des connaissances acquises depuis des années ? Ce serait une perte de temps et d'efficacité. Heureusement qu'il ne fonctionne pas de cette façon. A chaque fois que l'être humain apprend quelque chose de nouveau, cette nouvelle connaissance est ajoutées à celles déjà acquises.

Pendant plusieurs années, l'entrainement des modèles de langage se faisait en partant à chaque fois de zéro. Cette façon de travailler était assez couteuse en terme d'utilisation de ressource mais aussi contre-productive. Pour cette raison, le traitement du langage naturel accusait un certain retard par rapport à d'autres domaines comme la vision machine qui fonctionnait déjà avec le principe d'ajustement des connaissances accumulées par la méthode du « *transfert de connaissance* » en utilisant les modèles pré-entrainés d'ImageNet [17] [GF12 ; HAP17 ; Men+17].

La possibilité d'opérer un transfert de connaissance en traitement du langage naturel avait été testé sans vraiment beaucoup de succès jusqu'à la publication par HOWARD et RUDER de leur travail *« Universal language model fine-tuning for text classification »* [HR18]. Dans celui-ci, ils montrent l'avantage d'utiliser un modèle pré-entrainé pour les tâches de classification de texte. La méthode qu'ils proposent, « *Universal Language Model Fine-Tuning (ULMFiT)* » est d'utiliser un modèle pré-entrainé, l'ajuster et utiliser le résultat pour effectuer la tache voulue.

Cette méthode, dont la figure 3.12 illustre l'exécution, se déroule en 3 étapes. Ces phases sont le pré-entrainement du modèle de langage, son ajustement au domaine spécifique et l'ajustement de la tâche de classification voulue. Dans les deux premières étapes, le travail a recours à l'« *AWD-LSTM* »[MKS17], une architecture LSTM composée de 3 couches.

Quelles sont les particularité de chacune des étapes ?

1. « *Pré-entrainement* » : Celle-ci se fait sur une quantité de données considérable. Cela permet au modèle d'avoir une idée assez large des subtilités du langage. Cette étape bien que coûteuse, ne doit être faite qu'une seule fois.
2. « *Ajustement du modèle* » A ce stade, on intègre les spécificités du domaine de la tâche de classification. Il peut s'agir des textes de droits, du commerce, de la médecine, etc. Étant donné l'utilisation du modèle pré-entrainé, cette phase sera plus rapide et durera moins

17. Accessible à l'adresse `https://www.image-net.org/`

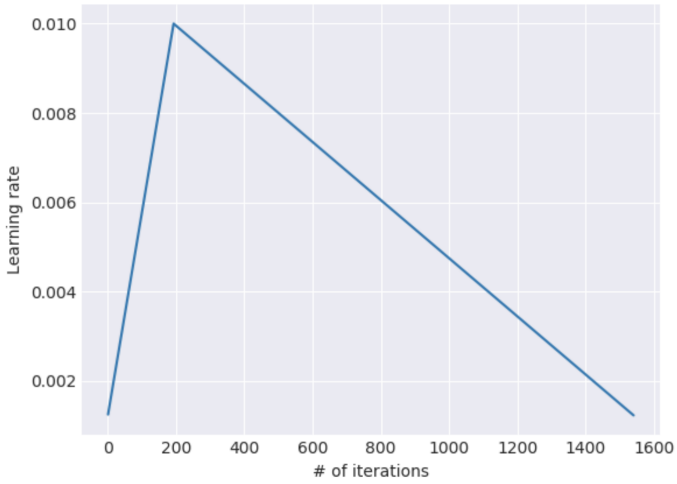


FIGURE 3.13 – UlMFiT : Courbe du taux d'apprentissage triangulaire oblique, fonction du nombre d'itérations [HR18]

longtemps que la précédente. Dans l'article, deux principes sont utilisés. Le premier est l'« *ajustements par discrimination* » [HR18] où un taux d'apprentissage différent est utilisé pour chaque couche. Soit $l = 1, 2, ..., L$ le numéro d'une couche. Le seul taux d'apprentissage à fixer est celui de la dernière couche η^L. Ensuite, le taux d'apprentissage des autres couches se calcule par $\eta^{l-1} = \eta^l/2.6$. Ce qui permet de ne faire l'ajustement que pour la dernière couche et celui-ci se propage aux couches précédentes. Pour chaque couche, la mise à jour des poids synaptiques se fait en utilisant la formule présentée dans l'équation (3.5) [HR18]. Dans cette dernière, $l = 1, 2, \ldots, L$ est le numéro de la couche, θ_t^l est le poids synaptique de la couche l à l'itération t et $\bigtriangledown_{\theta^l} J(\theta)$ le gradient.

$$\theta_t^l = \theta_{t-1}^l - \eta^l . \bigtriangledown_{\theta^l} J(\theta) \tag{3.5}$$

Le deuxième est de ne pas garder le taux d'apprentissage constant. Dans les premières itérations, il augmente linéairement de façon rapide. Ensuite, il diminue graduellement à chaque passage. C'est le principe du « *taux d"apprentissage triangulaire oblique* » [HR18], appellation probablement inspirée par la courbe de cet apprentissage comme illustré dans la figure 3.13.

3. « *Ajustement de la tâche de classification* » : Dans cette dernière étape, le modèle est entrainé en utilisant 2 blocs linéaires supplémentaires. La fonction d'activation *ReLu* est utilisée pour la couche intermédiaire et la *Softmax* pour la dernière. Chacune des bloc utilise le drop-out et la normalisation.

Grâce à ce transfert de connaissance et l'utilisation d'un modèle pré-entrainé, les modèles basés sur ULMFiT présentent l'avantage de permettre d'atteindre de très bon résultats même lorsque les données disponibles ne sont pas nombreuses. Ce transfert de connaissance a aussi permis l'avènement d'autres modèles qui s'en sont inspirés comme le GPT [Rad+18] et ses évolutions [Rad+19 ; Bro+20], BERT [Dev+18] et ses dérivées [Liu+19 ; Mar+19], etc.

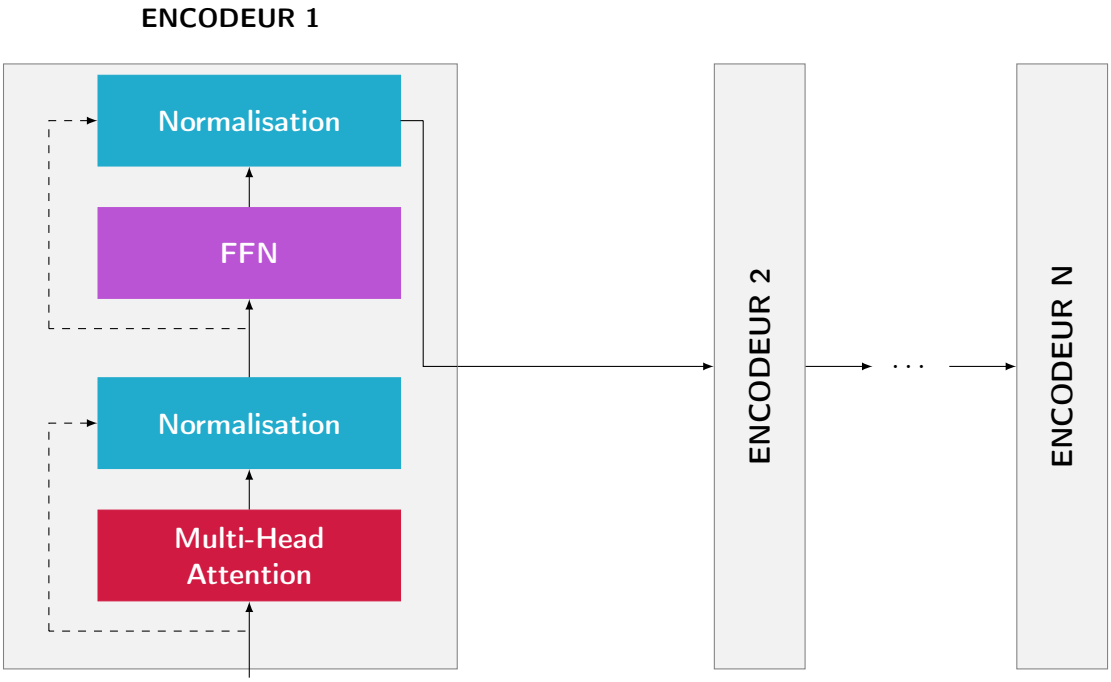


FIGURE 3.14 – Architecture simplifiée de BERT et ses variantes. Il faut noter que, selon le modèle, le nombre de blocs d'attention, la dimension de l'espace vectoriel ainsi que la répétition du nombre d'encodeurs varie

6.2 Bert et ses variantes

BERT acronyme de « *Bidirectional Encoder Representations from Transformers* » est un modèle de langage proposé par DEVLIN et al. dans l'article « Bert : Pre-training of deep bidirectional transformers for language understanding » [Dev+18]. Comme son nom l'indique, son architecture repose sur celle de transformeurs expliquée dans la section 5.3.3 avec la particularité de n'utiliser que le bloc encodeur. Tel qu'illustrée dans la figure 3.14, l'architecture est composée d'une succession de blocs encodeurs. Ce qui implique un dimension de l'espace vectoriel plus grand (784 pour BERT Base et 1024 pour BERT large) et plus de blocs [*Attention, Normalisation, FFN, Normalisation*].

L'utilisation du bloc encodeur des transformeurs rend BERT bidirectionnel. En effet, dans le bloc encodeur le vecteur d'attention d'un mot est calculé sur base de tous les mots de la phrase ce qui permet à BERT de déterminer un mot dans les deux sens. Tout le contraire du bloc décodeur, utilisé entre autre dans GPT [Rad+19], où le vecteur d'attention d'un mot est calculé sur base des mots qui le précède uniquement le rendant ainsi unidirectionnel.

Se basant sur le principe de transfert de connaissances comme pour ULMFiT, pour entrainer le modèle, BERT commence par une phase de pré-entrainement plus globale suivi de l'ajustement du modèle pour les tâches particulières. La première phase est composée de deux tâches à savoir le « *Masked language modeling (MLM)* » et prédiction de la phrase suivante « *NSP*[18] ». Quant à la seconde phase, elle s'exécute avec la même architecture que la première, à la différence

18. Next sentence prediction [Dev+18]

qu'elle va se focaliser sur des tâches spécifiques telles que la classification, la NER, l'inférence du langage naturel (NLI [19]), etc. Voyons comment se déroulent ces étapes.

6.2.1 Pré-entrainement

Lors de cette phase, pour la première tâche qu'est le MLM, on masque certains mots dans le but de permettre au modèle de prédire des mots sur base d'autres mots déjà présents de la phrase. Dans cette procédure, 15% de mots sont choisis de façon aléatoire. Une fois choisis, 80% de fois les mots sont masqués, 10% de fois certains mots sont remplacés par d'autres mots du corpus choisis de façon aléatoire et 10% de fois ils sont gardés tels quels. En prenant un exemple inspiré de l'article [Dev+18], pour la phrase `le trajet a été long`, le fonctionnement du masquage pour le mot `long` serait le suivant :

— 80% de fois, remplacer le mot `long` par `[MASK]` pour donner `le trajet a été [MASK]`,
— 10% de fois remplacer le mot `long` par un mot choisi au hasard, par exemple `le trajet a été maison`,
— 10% de fois ne rien changer à la phrase en gardant `le trajet a été long`.

Pour la seconde tâche qu'est le NSP, il est question de classification binaire dont l'objectif est de prédire si étant donné une première phrase, la deuxième est un successeur possible dans le corpus. La moitié du temps, la bonne phrase successeur est fournie et dans l'autre moitié une phrase choisie aléatoirement dans le corpus est fournie. Ce qui permet au modèle de s'adapter à un entrainement pour des tâches basées sur plusieurs phrases.

Il faut noter le temps considérable que nécessite le pré-entrainement de ce modèle. Mais, cette consommation en ressources est à relativiser étant donné que ce pré-entrainement n'est censé être effectué qu'une seule fois.

6.2.2 Ajustement

Lors de l'ajustement, le modèle de langage produit dans la première étape est utilisé et ajusté pour intégrer les subtilités d'une tâche donnée. BERT a montré son efficacité dans 11 des tâches évaluant la compréhension du langage (GLUE) tel que définie dans l'article *« GLUE : A multi-task benchmark and analysis platform for natural language understanding »* [Wan+18].

La phrase d'ajustement est moins gourmande en ressources et ne demande que quelques jours en utilisant un simple GPU et quelques heures en ayant recours un TPU [20]. Ce qui démontre ici encore l'avantage du transfert de connaissance dans la création des modèles de langages.

6.2.3 Les variantes de BERT

Après la sortie de BERT, plusieurs modèles ont vu le jour en utilisant la même architecture mais en adaptant certains de ses paramètres. En effet, l'objectif de ces modèles était de répondre à certaines faiblesses que présentait BERT. Celles-ci sont les suivantes :

— L'entrainement demande trop de ressources et de temps
— Un trop grand nombre de paramètres
— Certains chercheurs considère la méthode d'apprentissage de BERT comme n'étant pas assez efficace
— etc.

19. Named language inference
20. Tensor Processing Unit

Ainsi, le modèle ALBERT [Lan+19], propose une solution liée au trop grand nombre de paramètres qui ralentissent l'apprentissage. Dans ce modèle, au lieu de 110 millions de paramètres dans BERT, on passe à 12 millions ce qui rend le modèle facilement utilisable dans le monde réel, facilite son déploiement et le rend plus rapide. Dans ce modèle, la prédiction du prochain mot (NSP [21]) a été remplacé et les détails sur son processus d'apprentissage sont largement discutés dans l'article *« Albert : A lite bert for self-supervised learning of language representations »*.

Un autre modèle RoBERTa pour « *Robust Optimized BERT pre-training approach* » [Liu+19] est le fruit du travail des chercheurs qui considéraient BERT comme n'ayant pas été assez entrainé ou pouvait mieux l'être. Tout d'abord ils proposent la modification de la tâche du *MLM* et en particulier le masquage. Pour eux, l'apprentissage est bien meilleur en masquant plusieurs mots dans une phrase. Ils considèrent aussi l'entrainement du *NSP* comme n'apportant pas de valeur ajoutée au modèle et l'excluent du processus. Il faut aussi noter l'utilisation d'une masse importante de données d'apprentissage, environ 160 Gigaoctet de données contre 16 pour BERT.

Un autre modèle, Bort [WP20], a été récemment proposé comme une version légère mais efficace de BERT.

A part les problèmes liés aux performances, à la taille ou à la méthode d'apprentissage, d'autres modèles dérivés ont été entrainés exactement comme BERT pour couvrir le besoin en modèles de langage dans d'autres langues. C'est le cas par exemple de « *CamemBERT* » [Mar+20] entrainé sur un corpus de la langue française.

6.3 GPT

Contrairement à BERT, discuté dans la section précédente qui utilise le bloc encodeur des transformeurs, le modèle « *Generative Pre-Training (GPT)* »[Rad+18] proposé par les chercheurs de l'entreprise Open AI [22] va utiliser le bloc décodeur de l'architecture transformeur comme illustré dans la figure 3.15. Il faut cependant noter un changement majeur dans ce dernier en particulier le retrait du sous-bloc « Multi-Head Attention » (voir la figure 2.5) qui prenait en compte le résultat de l'encodeur.

Il est aussi important de noter la différence entre « *Self-Attention* » et « *Masked-Self Attention* ». Dans le premier, le vecteur d'attention d'un mot est créé sur base de tous les mots de la phrase, alors que dans le dernier il est calculé en ne se basant que sur les mots le précédent.

La procédure d'entrainement du modèle se compose, tout comme BERT, de deux phases. « *La phase de pré-entrainement consistant à l'apprentissage du modèle de langage sur une quantité de donnée considérable suivie de la phase d'ajustement où le modèle est adapté à une tâche en utilisant les données annotés* » [Rad+18]. Quel objectif est recherché au cours de ces deux phases ?

6.3.1 Pré-entrainement

Cette phase utilise l'apprentissage non supervisé. L'objectif recherché est la maximisation de la fonction (3.6) [Rad+18]. En d'autres mots, considérant le corpus $\mathcal{U} = u_1, \ldots, u_n$, maximiser la probabilité que le mot u_i fasse partie du contexte des mots $u_{i-k}, \ldots, u_{i-1}$. Avec k la taille de la fenêtre des mots qui précèdent et Θ les paramètres du modèle.

21. Next Sentence Prediction
22. `https://openai.com`

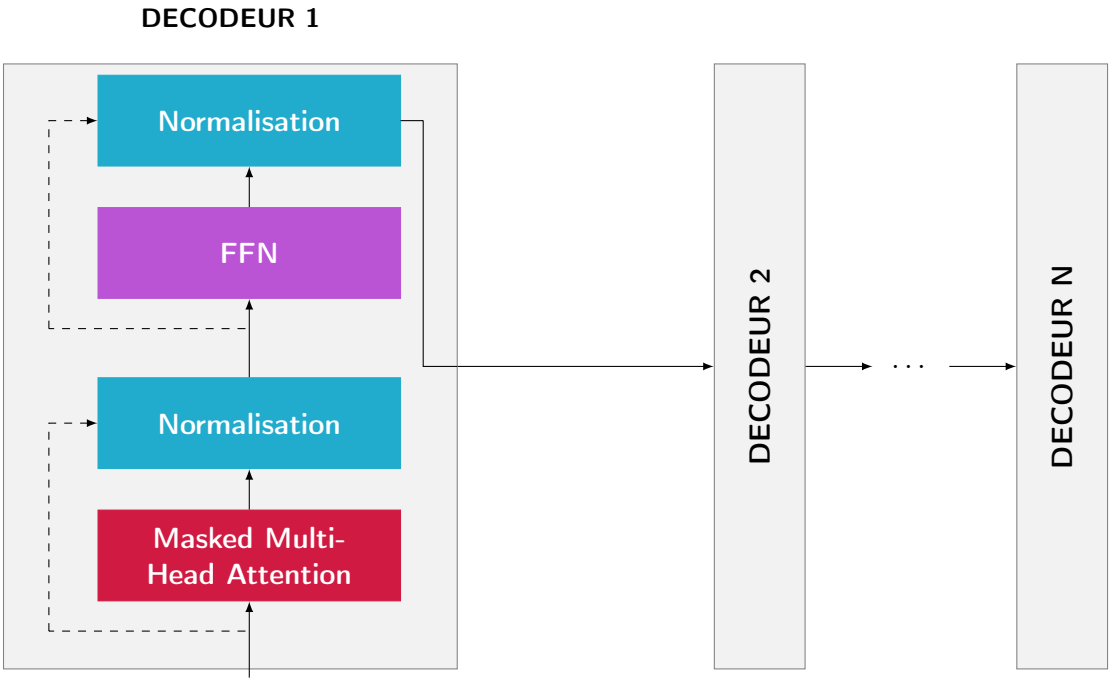


FIGURE 3.15 – Structure simplifiée de GPT

$$L_1(\mathcal{U}) = \sum_i \log\, P(u_i|u_{i-k}, \ldots, u_{u-1}, \Theta) \tag{3.6}$$

6.3.2 Ajustement

Dans cette phase, l'objectif étant d'entrainer le modèle pour des tâches spécifiques, un apprentissage supervisé est utilisé. L'objectif recherché est de maximiser la probabilité d'observer une annotation y étant donné des mots $x_1, \ldots, x_n$. Cet objectif peut être exprimé par la fonction (3.7) où C correspond aux données annotées pour les besoins de l'entrainement et chaque élément de $\mathcal{C}$ est composé d'une séquence de mots avec l'annotation y correspondante.

$$L_2(C) = \sum_{x,y} \log\, P(y|x^1, \ldots, x^m) \tag{3.7}$$

En plus, plutôt que simplement maximiser l'équation (3.7), les auteurs ont ajouté un objectif d'apprentissage complémentaire pour arriver à l'objectif final de l'équation (3.8) avec λ le poids de cet objectif.

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda L_1(\mathcal{C}) \tag{3.8}$$

De plus, lors de cette phase, deux couches sont ajoutées à l'architecture transformeur. L'une munie d'une fonction d'activation linéaire et l'autre d'une Softmax.

6.3.3 De GPT à GPT-3

La toute première version de GPT était composée de 12 blocs de décodeurs et avait au total 117 millions paramètres. Cette version avait été entrainée en utilisant le « *BookCorpus* » [23] composé d'environ 74 millions de phrases et 1 milliard de mots [Zhu+15].

Ensuite, dans GPT-2 [Rad+19] l'objectif est d'apprendre plusieurs tâches en utilisant le même modèle. Normalement l'objectif d'entrainement du langage serait la probabilité d'avoir une sortie donnée considérant une entrée. Cet objectif d'entrainement va donc devoir changer pour devenir la probabilité d'avoir une sortie donnée, considérant une entrée et une tâche. C'est ce qui va être appelé « *conditionnement par tâche* », idée selon laquelle un modèle va produire pour le même entrée une sortie différente en fonction de la tâche. L'autre changement dans GPT-2 est sa capacité à comprendre l'instruction d'une tâche donnée sans qu'aucun exemple ne lui ait été fourni. Il est entrainé en utilisant encore plus de données (40 G). Il faut aussi noter que dans cette deuxième version, la structure du modèle est composée de 96 couches et comporte plus de 1,5 milliard de paramètres, soit 10 fois plus que son prédécesseur.

Continuant sur sa lancée, Open API développe une troisième version le modèle GPT-3 [Bro+20]. Il a 100 fois plus de paramètres que son prédécesseur. De par l'aveux de ses concepteurs, ce modèle est considéré comme présentant potentiellement un risque de mauvaise utilisation à cause de ses capacités de génération de texte. Il pourrait être utilisé par exemple pour générer des fausses informations, du contenu indésirable. De plus, la qualité des textes ainsi généré est emprunt de préjugés en fonction du texte utilisé pour l'entrainement.

Bien que GPT-3 présente ces capacités impressionnantes, il est sous licence propriétaire et le modèle n'est pas accessible au public. Cependant, une version open source disposant de 6 milliards de paramètres, GPT-J [WK21], a été entrainée et rendue public par *Eleuther AI*[24].

7 Et maintenant ?

Ce chapitre nous a permis d'acquérir les bases théoriques nécessaires au développement d'un modèle de langage. Pendant des années, la création de ces modèles demandait à chaque fois de commencer l'entrainement du modèle à zéro, jusqu'à ce que l'utilisation des réseaux de neurones et du mécanisme de transfert de langage ait montré son efficacité.

Aujourd'hui, de nombreux modèles existent et peuvent permettre de mener à bien la tâche visée par ce travail à savoir l'élaboration d'une intelligence munie d'une capacité de compréhension et de classification des documents juridiques. Avec pour but final de pouvoir procéder à la prédiction de décision de justice.

Avec toutes les informations réunies jusqu'ici, nous pouvons à présent nous atteler à l'entrainement d'un modèle juridique. Celui-ci aura pour but de déterminer la recevabilité ou non d'un pourvoi en appel d'une décision judiciaire.

23. 5Go de données téléchargeables depuis l'adresse `https://huggingface.co/datasets/bookcorpus`
24. `https://eleuther.ai/`

CHAPITRE 4

Développement de la solution

1 Introduction

En ayant recours aux différents aspects théoriques abordées dans les chapitres précédents, nous allons à présent aborder le développement de la solution dont nous essayerons de vérifier la faisabilité.

Il aurait été intéressant de comparer les trois modèles discutés dans le chapitre 3, à savoir ULMFIT, une variante de BERT et une version de GPT pour voir quelle en est l'efficacité dans notre cas d'utilisation. Cependant, ne disposant pas d'une puissance de calcul suffisante, nous allons utiliser la méthode ULMFiT pour l'entrainement et la création de notre modèle.

Nous allons dans un premier temps commencer par définir l'étendu de notre travail et plus particulièrement ce à quoi nous comptons aboutir au terme de ce chapitre. Dans un deuxième temps, nous présenterons les outils utilisés pour implémenter notre solution. Ensuite nous parcourrons, les différentes étapes qui composent la réalisation de ce travail, en partant du choix de données à utiliser pour l'entrainement de notre modèle jusqu'à la réalisation de celui-ci à proprement parler.

Ce parcours sera suivi par un aperçu des différentes obstacles que nous avons rencontrés lors du développement. Nous en profiterons aussi pour expliquer les choix qui ont été faits.

2 Définition de l'étendue

Dans le premier chapitre concernant le droit nous avons établi un point spécifique où la solution que nous proposons peut jouer un rôle significatif. C'est dans cette optique que nous allons définir l'étendue de notre projet.

En effet, notre postulat est qu'il est possible de procéder à l'entrainement d'un modèle de classification capable de prédire si l'appel d'une décision de justice rendu dans une cour inférieure est recevable ou non. Nous avons aussi, dans le premier chapitre, identifié la cour d'appel civile comme étant celle où notre solution aurait du sens pour essayer d'en réduire le nombre des affaires pendantes. Le modèle que nous souhaitons développer devra avoir la capacité de faire ces prédictions sur les décisions prises dans les tribunaux de première instance.

Pour réaliser notre modèle, nous allons avoir recours à l'intelligence artificielle et au traitement du langage naturel, deux sujets largement abordés dans les deuxième et troisième chapitres de ce travail. Nous allons développer un modèle de langage en utilisant la méthode « ULMFIT » décrite dans la section 6.1. L'avantage que nous trouvons à cette méthode est la possibilité d'entrainer notre modèle et de l'enrichir après coup sans avoir recours à beaucoup de ressources.

L'entrainement de ce modèle de classification se fera en trois étapes, le pré-entrainement, l'ajustement du modèle de langage au domaine spécifique et l'ajustement de la tâche de classification. Comment allons nous nous y prendre particulièrement dans notre cas ?

2.1 Pré-entrainement

A ce stade nous n'allons pas nous occuper de cette phase d'entrainement car elle est trop couteuse à réaliser. Nous allons à la place essayer de trouver un modèle pré-entrainé pouvant être utilisé. Il devra s'agir d'une modèle adaptée à la langue cible, qui dans notre cas est le français. Recourir à un modèle existant et capable de comprendre le français, nous permettra de réduire le temps nécessaire à l'entrainement du modèle en ce qui concerne notre discipline cible, à savoir la compréhension, du point de vue linguistique, des textes de jurisprudence. Au terme de cette étape, il sera possible pour notre modèle, en lui fournissant un début de phrase, de prédire le mot suivant. L'objectif est d'atteindre un taux de précision d'environ 40%.

2.2 Ajustement du modèle au domaine

Muni de notre modèle pré-entrainé et des différents textes de jurisprudence, nous allons ajuster le premier modèle en utilisant ces derniers pour produire un modèle capable de comprendre un texte juridique. Le modèle de langage que nous enrichirons est capable de compléter le début d'une phrase en français. Dans cette même optique, le modèle que nous entrainerons au terme de cette étape, devra être capable de compléter le début d'un texte de jurisprudence.

2.3 Ajustement de la tâche de classification

A ce stade nous allons utiliser le modèle entrainé et décrit dans la section 2.2 pour l'enrichir et lui permettre de réaliser notre tâche de classification. Cette étape est celle qui permettra à notre solution de prédire la recevabilité ou non de l'appel d'une décision de justice rendu en première instance. Nous nous focaliserons sur l'aspect analyse de sentiment, non au sens sentimental du terme, mais plutôt dans le sens de savoir si l'issue sera négatif dans le cas où le jugement du tribunal de première instance est confirmé ou positif en cas d'annulation partielle ou totale de celui-ci.

Pour mener à bien cette étapes nous aurons besoins des données suivantes :

1. Les décisions de justice prises au niveau de la cour d'appel ainsi que l'issue de celles-ci. Cette dernière information sera utile car c'est elle que nous utiliserons comme résultat attendu.
2. Les décisions de justice prise en première instance seront d'un importance capitale car notre objectif est qu'une décision de justice passée en entrée de notre modèle puisse permettre de prédire le résultat de son appel. Raison pour laquelle dans notre cas l'issue de la décision n'a pas vraiment d'importance.

Pour que ces données soient utilisables, nous aurons en plus de celles énumérés ci-dessus, besoin de faire le lien entre les décisions de première instance et celles des cours d'appel.

3 Outils utilisés

Pour développer cette solution nous allons avoir recours à des outils nous permettant d'exécuter ces tâches de façon efficace. Nous avons généralement eu recours à des outils éprouvés, gratuits et open source.

3.1 Langage de programmation et libraires

Nous utiliserons le Python, en particulier sa version 3.8, comme langage de programmation pour développer notre modèle. Ce choix a été guidé par notre expérience professionnelle dans son utilisation. Notre objectif était aussi de ne pas réinventer la roue et d'utiliser les librairies déjà existantes et qui ont été spécifiquement développées pour les tâches qui nous intéressent.

3.1.1 Beautiful Soup 4

Cette librairie facilite le traitement des contenus XML et HTML. Nous l'utiliserons pour la constitution de notre data set de jurisprudences. Elle nous sera particulièrement utile pour ne procéder à la récupération que de quelques portions de textes dont nous avons besoin sur les pages proposant la jurisprudence.

3.1.2 Fastai

Cette librairie est décrite par ses auteurs comme étant « *une façon simple d'entrainer les réseaux neuronaux efficaces en utilisant les pratiques modernes* ». Son utilisation est largement abordée par le livre *Deep Learning for Coders with Fastai and Pytorch : AI Applications Without a PhD* [HG20].

Cette librairie, développée en sur-couche de PyTorch [1], donne la possibilité de procéder à l'entrainement de réseaux neuronaux dans différents domaines d'application comme la vision et le traitement du langage naturel et est nativement intégrée à un autre service, *Google Colab* (voir la section 3.2), que nous comptons utiliser. L'autre avantage est la possibilité de facilement modifier son comportement et sa configuration. Dans ce travail, nous nous limiterons à une utilisation simplifiée.

3.1.3 Autres librairies

Du reste, d'autres librairies nous seront utiles. Elles font pour la plupart toutes parties des librairies standard python. Ce sont les suivantes :

— *requests* : Pour exécuter des api ou télécharger les contenus des pages web.
— *os* : Pour accéder au système et y effectuer des opérations de vérifications
— *pathlib* : Pour une création des dossiers et un accès simplifié à ceux-ci et à leurs contenus.
— *multiprocessing* : Pour paralléliser le traitement de certaines fonctions.
— *tarfile* : Pour la décompression des fichiers au format tar.gz

3.2 Google Colab

Après avoir tenté de localement effectuer certaines tâches, il est apparu que la machine que nous avions à disposition n'avait pas assez de ressources pour procéder à l'entrainement de notre

1. Voir l'adresse `https://pytorch.org/`

réseau de neurones. En faisant quelques recherches nous sommes tombés sur Google Colab [2], un service gratuit mis à disposition par Google pour un temps limité. Ce service supporte *Python* et permet d'exécuter du code python depuis le navigateur.

En plus de supporter Python, il permet aussi d'avoir accès aux ressources auxquelles il aurait était inimaginable d'accéder autrement. Ce qui est intéressant avec ce service, c'est la capacité d'accéder à du GPU gratuitement ce qui facilite l'entrainement d'un modèle.

Il est à noter que les limites sur les ressources disponibles ne sont pas bien définies, mais lors de notre travail nous avons remarqué que lors de son utilisation, la machine virtuelle mise à disposition était configurée avec 70 Go d'espace de stockage et 16 Go de mémoire. De plus, cette machine était mise à disposition pour un temps limité de 24 heures.

3.3 Kaggle

En avançant dans notre tâche, nous avons aussi trouvé un autre Service, Kaggle [3], qui permet d'avoir accès aux ressources nécessaires à l'entrainement gratuitement.

Par défaut nous avions accès à une machine virtuelle pour une durée de 12 heures configurée avec 70 Go d'espace de stockage dont 50 temporaires, 16 Go de mémoire, et 4 CPU. Moyennant la validation de son compte par SMS, nous pouvions avoir accès de la GPU disposant de 16Go de mémoire. Il est à noter que l'utilisation du GPU entraine la réduction du CPU disponible passant de 4 à 2.

Le plus intéressant avec Kaggle, c'est la définition dès le départ du temps GPU mis à disposition. Ce dernier est fixé à 41 heures par semaine, largement assez pour effectuer l'entrainement de nos différents modèles.

4 Jeu de données

Une fois notre boite à outil déterminée, il est temps de trouver le jeu de données dont nous avons besoin. Notre objectif n'est pas d'à chaque fois procéder aux différentes tâches définies plus tôt. Raison pour laquelle, nous allons essayer autant que possible de trouver un modèle pré-entrainé ayant une compréhension de la langue française. Nous l'enrichirons pour le rendre capable, en plus de comprendre le français, d'acquérir notre domaine cible. En d'autres mots qu'il soit capable de comprendre le langage juridique. Pour rappel, quand nous parlons compréhension, nous faisons référence à la capacité du modèle à prédire le mot qui suit un mot donné. Sans ce modèle, il est impossible de réaliser un modèle de langage efficace adapté à la jurisprudence. Ensuite, nous allons procéder à la constitution du jeu de données pour l'ajustement et celles permettant l'entrainement à la tâche de classification.

4.1 Modèle de langage pré-entrainé

Nous avons commencé notre recherche via la librairie Fastai. En effet, celle-ci propose des modèles pré-entrainés pouvant être utilisés. Malheureusement nous n'y avons pas trouvé de modèle de langage pré-entrainé en français. Ensuite, nous avons parcouru les différents projets utilisant la méthode ULMFIT dans l'espoir de trouver un modèle pré-entrainé en français. Il s'est avéré que trouver un tel modèle était impossible.

2. Accessible à l'adresse `https://colab.research.google.com/`
3. Accessible à l'adresse `https://www.kaggle.com/`

Suite à ce constat, nous n'avons d'autre choix que de procéder nous même au pré-entrainement, en espérant avoir des ressources suffisantes pour procéder à un tel exercice. Encore faut-il disposer d'un jeu de données pour réaliser cette tâche. Mais quelles données allons nous utiliser ?

4.2 Données d'entrainement du modèle de langage

Comme pour la réalisation des différents modèles pré-entrainé proposé par Fastai, nous avons décidé d'avoir recours aux données de la version française de Wikipédia. Heureusement il est possible de télécharger toutes les données texte de Wikipédia dans la langue qui nous intéresse.

Nous avons donc procédé au téléchargement des données. Tous les articles sont dans un XML dont la taille de l'archive est d'environ 5,5 Go pour la version compressée et 25 Go une fois décompressé.

Il est difficile de travailler avec un fichier d'une aussi grande taille et au format xml de surcroit. Nous allons donc commencer par découper ce gros fichier en articles. Le contenu de chacun d'eux se trouve dans une balise doc (`<doc␣*>(.*)</doc>`).

Lors du traitement de chaque document, nous allons effectuer les opérations suivantes :

1. Supprimer les balise `doc`.
2. Supprimer la première ligne : Dans cette archive la première ligne correspond au titre du document. Or ce qui nous intéresse ce sont des phrases compètes et non les mots indépendants.
3. Supprimer les espaces en début et en fin de texte.
4. Ne garder que les textes faisant plus de 1 800 caractères. Cette condition nous permet de ne pas garder certains textes qui n'apporteraient rien d'autre que du bruit à notre jeu de données.

Pour préparer ces données et les rendre facilement utilisables, nous avons commencé par passer le fichier XML dans un outil, Wikiextractor [Att15], dont la tâche principale a été de retirer la plupart des bruits présents dans le fichier initial. Dans notre cas nous sommes passé d'un fichier XML de 25 Go à un de 5,68 Go ne contenant que les articles présents dans l'export initial.

Nous avons ensuite procédé à la découpe de ce fichier et avons sauvegardé chaque article dans un fichier pour en faciliter l'utilisation par la suite. Deux dossiers ont ainsi pu être constitués, l'un contenant l'ensemble des articles, et l'autre n'étant composé que par ceux disposant d'un nombre de caractères supérieur à 1 800.

Ensuite nous avons créé trois jeux de données, dont un premier composé 28595 articles, soit l'équivalent de ce qui avait été utilisé dans l'article publié par Howard et Ruder [HR18]. Les détails sur l'ensemble des jeux de données constitués sont repris dans la table 4.1.

Jeu de données	Nombre d'articles	Nombres de mots
Wiki-28k	28 595	27 092 983
Wiki-100k	100 000	99 461 381
Wiki-500k	574 674	566 153 989

Table 4.1 – Jeux de données constitués sur base des articles francophones de Wikipédia. Seuls ceux composé d'au moins 1 800 caractères ont été conservés.

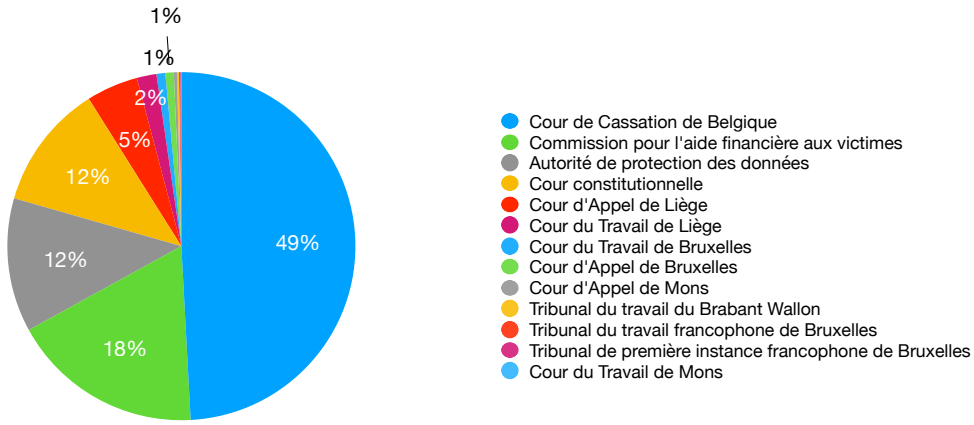


FIGURE 4.1 – Publications des décisions en français par cour, tribunal ou commission entre janvier 2019 et décembre 2021. En pourcentage d'un total de 4 873.

A ce stade nous avons les données nécessaires à la création d'un modèle pré-entrainé. Il est temps, de voir ce qu'il en est des données de jurisprudence.

4.3 Documents juridiques

Dans la liste des documents juridiques, il est important de différencier ce dont nous avons besoin. D'une part, il nous faut des données juridiques pour ajuster le modèle de langage pré-entrainé à la compréhension de la jurisprudence, ensuite nous devons établir un jeu de donnée pour procéder à l'ajustement de la tâche de classification.

4.3.1 Pour l'ajustement du modèle pré-entrainé

En Belgique, les données de jurisprudence sont normalement censées être mises à disposition du public. Un portail, *Juportal*[4], existe pour en permettre la consultation. Cette plateforme offre un moteur de recherche. Cependant il a été très décevant de se rendre compte que seul très peu de décisions récentes y sont publiées. En effet, comme illustré dans la figure 4.1 nous avons remarqué qu'en cherchant les décisions en français, sur les trois dernières années[5], 4 873 décisions y ont été publiées dont seules 435 par les cours d'appel et les tribunaux de première instance soit environ 8,93 % des décisions, ce qui est négligeable comparé aux dossiers traités annuellement par ces cours et tribunaux.

Il s'est avéré que même si ces données étaient disponibles pour consultation, il n'existait aucune API permettant d'y accéder facilement. Ce qui nous a été confirmé, par le service responsable du portail, par courrier électronique. Heureusement, il existe des initiatives comme

4. Plateforme de consultation des décisions de justice accessible à l'adresse `https://juportal.be/home/accueil`

5. Période allant du 01/01/2019 au 31/12/2021

Open Justice[6]. Leur projet Omdat[7] expose une API permettant de consulter ces données. C'est ce projet que nous utiliserons pour établir notre jeu de donnée pour la simple et bonne raison qu'elle permet de pouvoir filtrer pour ne récupérer que les décisions dans la langue qui nous intéresse.

Pour constituer un jeu de données conséquent permettant la création d'un modèle capable de comprendre le langage juridique belge en français, il nous est apparu que l'expression juridique n'a pas beaucoup changé les 20 dernières années et que même si les décisions ont pu changer suite à l'évolution de la loi, leur rédaction était elle restée la même.

En établissant ces données nous avons rencontré un problème majeur. Parfois certaines décisions étaient référencées mais le texte du document n'était pas disponible. Nous avons alors dû procéder à la récupération des textes directement depuis Juportal. Pour éviter ce genre de problème nous utilisons Omdat pour récupérer les ID au format *ECLI*[8] et filtrer notre recherche. Ensuite grâce à ces id, nous récupérons les textes dans le cas où ceux-ci sont présents sur Juportal.

Dans cette phase, nous utilisons Beautiful Soup pour analyser le contenu des pages Juportal et récupérer le texte présent dans la balise HTML portant l'ID `plaintext`. Nous ajoutons le texte à notre corpus s'il n'est pas vide. Dans certains cas le contenu du texte se limitait à <>. Ces décisions seront aussi exclues de notre corpus.

Après tous ces ajustements, nous avons pu constituer un jeu de donnée composé d'environ 19 006 documents pour 56 615 764 mots.

4.3.2 Pour la classification

Notre objectif est de prédire si l'appel d'une décision de justice rendu en première instance sera recevable ou non en appel. Pour aboutir à notre résultat, nous comptons ne recourir qu'à deux paramètres.

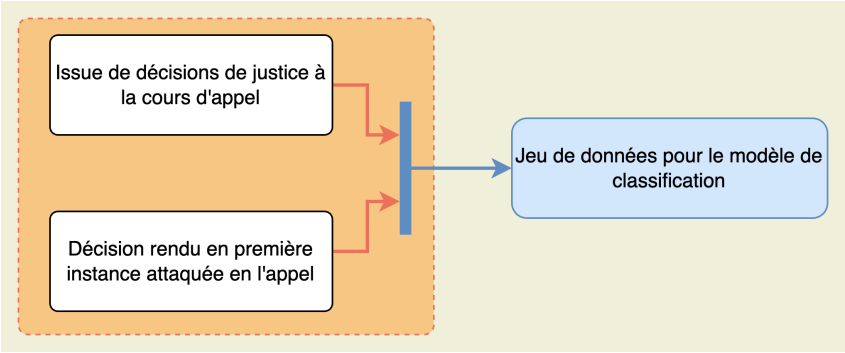


FIGURE 4.2 – Classification : Processus de constitution du jeu de données

Comme illustré dans la figure 4.2, nous avons besoin d'une part des décisions de justice rendues en première instance et d'autre part, pour celles dont un appel a été introduit, l'issue du dossier en appel. Notre travail doit dès lors être fait dans le sens inverse. Il nous paraît

6. Une association dont l'objectif est de rendre le droit plus accessible et ce compris les décisions de justice. Site web accessible à l'adresse `https://openjustice.be`

7. Projet permettant de parcourir la jurisprudence belge et accessible à l'adresse `https://omdat.openjustice.lltl.be/`

8. European Case Law Identifier

logique qu'il faut tout d'abord trouver les jugements rendus en appel et de ceux-ci remonter aux décisions des cours inférieurs leur correspondant.

Concernant l'issue des jugements, contrairement à la France où chaque décision de justice [9] publiée est accompagnée de son issue ; en Belgique à moins de procéder à la lecture complète du jugement aucune indication concernant l'issue n'est fournie. Mais celui-ci peut facilement être trouvé en se basant sur les mots clés généralement utilisés comme `non␣reçu`, `rejetté`, `recevable`, ...

Par contre, après l'analyse des différents documents de jurisprudence collectés, il est impossible de trouver dans les jugements en appel l'indication de la décision attaquée. La seule piste d'information parfois fournie est le nom du tribunal et la date de la décision de justice ainsi qu'un identifiant dont le format n'est pas fixe.

En gros nous nous trouvons face à un mur concernant un possible accès à ces données. En effet, notre contact avec Juportal n'a pas été fructueux, nous avons été renvoyé au Disclaimer du portail par les équipes qui en assurent la gestion.

Malgré cette difficulté, nous allons tout de même procéder à l'entrainement de deux premiers modèles et plus tard prendre une décision concernant le modèle de classification qui, à se stade, semble être impossible à entrainer.

5 Modèle pré-entrainé

En utilisant les données constituées dans la section 4.1, nous allons à présent procéder à cet entrainement. Dans un premier temps, nous utiliserons le jeu de donnée Wiki-28k pour créer ce modèle pré-entrainé sans lequel nous ne pouvons pas aller plus loin dans notre travail.

C'est à cette étape que les différentes classes et fonctions, mais aussi les algorithmes fournis par la librairie Fastai vont nous être utiles. La réalisation de cette tâche passera par les étapes suivantes :

1. Définition du tokeniseur,
2. Définition du *Data block*,
3. Définition des *DataLoaders*,
4. Création de *l'instance d'apprentissage*,
5. Apprentissage du modèle et,
6. Sauvegarde du modèle.

5.1 Définition du tokeniseur

Nous allons utiliser le SentencePieceTokeniser pour procéder à la tokenisation de notre jeu de donnée. Ce tokeniseur utilise SentencePiece[KR18] qui est un modèle de tokenisation efficace des langues même celles non européennes. Son utilisation nécessite l'installation de la librairie python sentencepiece [10].

9. Voir l'exemple disponible à l'adresse `https://www.legifrance.gouv.fr/juri/id/JURITEXT000046090045`

10. Le code source de cette librairie est disponible sur Github à l'adresse `https://github.com/google/sentencepiece/`

	text	text_
0	__xxbos __xxmaj __le __ rif t __de __xxmaj __greg ory __est __la __branche __orientale __du __ rif t __est - africain . __xxmaj __la __faille __est __consécutive __à __l ' éloignement __de __la __plaque __som al ienne __et __de __la __plaque __ africaine __(pla que __n ub ienne __sur __le __schéma __ ci - jo int). __xxmaj __bien __que __le __terme __soit __parfois __employé __dans __un __sens __restreint __pour __désigner	__xxmaj __le __ rif t __de __xxmaj __greg ory __est __la __branche __orientale __du __ rif t __est - africain . __xxmaj __la __faille __est __consécutive __à __l ' éloignement __de __la __plaque __som al ienne __et __de __la __plaque __ africaine __(pla que __n ub ienne __sur __le __schéma __ ci - jo int). __xxmaj __bien __que __le __terme __soit __parfois __employé __dans __un __sens __restreint __pour __désigner __le

FIGURE 4.3 – Un exemple de jeu de donnée tokenisé

5.2 Définition du Data block

Dans Fastai le *data block* permet de configurer la façon dont nous voulons utiliser les données dans les *Data Loaders*. C'est au datablock que nous passons les définitions initiales concernant notre jeu de donées et c'est lui qui s'occupera de la tokenisation des données. Nous lui informons des aspects suivants :

— Le tokeniseur à utiliser,
— S'il s'agit d'un modèle de langage ou non
— La méthode à utiliser pour la récupération des données, `get_files` dans notre cas, et
— La façon dont nous voulons utiliser les données. Dans notre cas nous allons faire recours à une découpe aléatoire où nous garderons 10% de nos données pour la validation.

5.3 Définition des Dataloaders

Ce sont les dataloaders qui se chargent de récupérer les données et d'en faire le découpage. Du datablock précédemment créé, nous récupérons les dataloaders. Grâce au dataloader nous pouvons avoir les informations sur l'état de notre jeu de données à ce stade. Nous pouvons par exemple savoir que le Datablock a établi 2 jeux de données faisant respectivement 25736 et 2859 le premier utilisé pour l'entrainement et le dernier pour la validation. Comme illustré dans la figure 4.3, nous pouvons aussi voir la façon dont nos données on été tokenisées.

5.4 Création du langage model learner

Nous allons à présent créer le langage model learner, notre instance qui va se charger de l'apprentissage. Nous lui précisons quelle architecture neuronale utiliser. Nous utilisons l'AWD-LSTM qui est celle utilisée par la méthode ULMFIT (voir dans la section 6.1). Sans oublier de spécifier au learner qu'il s'agit d'un premier apprentissage.

Avant d'entrainer notre modèle, nous allons commencer par trouver le taux d'apprentissage idéal. Un taux sera considéré idéal aussi longtemps que notre courbe est décroissante. Grâce au graphique de la figure 4.4 nous en déduisons une valeur de taux idéal appartenant à l'ensemble $[10^{-3}, 10^{-2}]$.

Une fois ce taux d'apprentissage défini nous pouvons à présent configurer notre modèle et lancer son apprentissage.

5.5 Apprentissage du modèle

En ce qui concerne le taux d'apprentissage, nous allons le fixer à $\frac{10^{-2}}{2}$ et nous réaliserons l'apprentissage en 10 sessions.

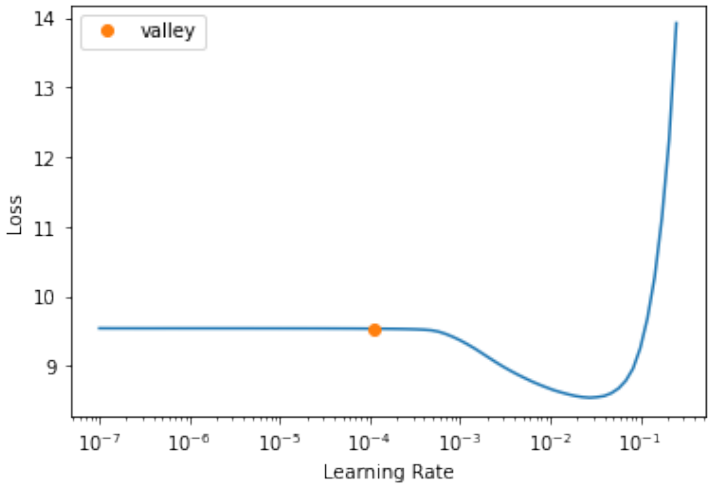


FIGURE 4.4 – Modèle de langage : Taux d'apprentissage par rapport à la perte

Epoch	Training loss	Validation loss	Accuracy	Perplexity	Time
0	3.986532	4.062321	0.281561	58.109035	29 :35
1	3.668345	3.714916	0.318371	41.055130	29 :10
2	3.429180	3.582272	0.332818	35.955132	29 :14
3	3.488193	3.488349	0.342594	32.731876	29 :17
4	3.350857	3.400212	0.353990	29.970455	29 :26
5	3.324047	3.318959	0.362740	27.631578	29 :21
6	3.244368	3.241565	0.372864	25.573700	29 :03
7	3.184911	3.169890	0.381687	23.804869	30 :06
8	3.129984	3.123524	0.388888	22.726336	30 :28
9	3.165579	3.111521	0.391341	22.455173	30 :08

TABLE 4.2 – Métriques relatives à la création de notre modèle de langage en français

La table 4.2 reprend les données relatives à cet entrainement. Dans la colonne *epoch*, nous avons le numéro de session, dans *training loss* la perte lors de l'entrainement, dans *validation loss* la perte lors de la validation de l'apprentissage, *accuracy* la précision de notre modèle et dans *perplexity* le degré d'incertitude de notre modèle lors de la prédiction du mot suivant et enfin dans la colonne time, la durée de l'apprentissage.

Après 10 sessions, nous avons réussi à créer un modèle de langage capable de prédire le mot suivant dans plus d'une fois sur trois.

Il est possible d'améliorer ce modèle. En effet, en analysant les données de la table 4.2, nous remarquons que les valeurs *validation loss*, *accuracy* et *perplexity* peuvent encore être améliorées. Pour cela il nous faudrait beaucoup plus de donnée ainsi que l'entrainement du modèle au delà des 10 sessions, mais cela nécessite d'avoir une quantité non négligeable de ressource à disposition. Cette amélioration est aussi possible par l'utilisation le jeu de données Wiki-100k pour l'entrainement, mais il est impossible de procéder à une telle amélioration par manque de ressources.

En tout, l'entrainement de ce modèle a durée 4 heures 55 minutes et 48 secondes. Ce qui est

relativement raisonnable et permet donc de vérifier qu'il est possible de procéder à l'entrainement d'un modèle de langage sans avoir besoin à recourir à une très grosse quantité de donnée. Ceci confirme aussi l'idée selon laquelle l'utilisation de la méthode ULMFIT et du modèle AWD LSTM en utilisant la librairie Fastai est justifiée.

5.6 Sauvegarde du modèle

Nous sauvegardons notre modèle pour une utilisation ultérieure. Nous procédons à la sauvegarde de deux fichiers distincts. D'une part sauvegarder le modèle de langage qui comprend la matrice représentant les mots sous forme de vecteurs. Chaque ligne correspondant à une ligne dans le vocabulaire. Raison pour laquelle nous devons, en plus du modèle de langage, sauvegarder le vocabulaire.

Le code utilisé pour procéder à l'entrainement de ce modèle est fourni dans l'annexe A. Muni du modèle sauvegardé, nous pouvons à présent procéder à l'étape suivante.

6 Ajustement du langage au domaine

L'ajustement du modèle de langage au langage spécifique, passe par les même étapes que pour le premier. A part quelques ajustements qu'il sera nécessaire d'appliquer, dans les deux cas nous utilisons les même étapes.

Les définitions du tokeniseur, du Data block et des Dataloaders restent identiques. Cependant la création de l'instance d'apprentissage sera totalement différente.

6.1 Instance d'apprentissage

Comme nous ne créons pas le modèle de langage en partant de rien, nous utilisons le modèle de langage généré dans l'étape précédente. Nous avons besoin à la fois du fichier contenant le modèle mais aussi du vocabulaire.

Nous gardons la même configuration pour ceux-ci dont entre autres le fait d'utiliser 10% de données pour la validation. Deux jeux de données sont crées. Composés respectivement de 17105 et 1900 documents, ils seront utilisés respectivement pour l'entrainement et pour la validation.

Nous avons aussi besoin de déterminer le taux d'apprentissage à utiliser. Nous déterminons ce taux grâce à la courbe représentée dans le graphique de la figure 4.5. En observant celle-ci, nous pouvons établir le taux d'apprentissage comme devant appartenir à l'ensemble $[10^{-2}, 10^{-1}]$.

Dans notre cas nous avons décidé d'utiliser un taux d'apprentissage de 10^{-2}.

Nous avons commencé par l'exécution d'une première session pour entrainer la dernière couche et en évaluer la précision. Il est encourageant de voir qu'au terme de celle-ci, le taux de précision initial du modèle d'ajustement est de le 46,27 % et l'incertitude est divisé par deux. Nous pouvons aussi remarquer une réduction conséquente de la perte lors de la validation (voir la figure 4.6) comparé au modèle de langage initial.

Après cette première session, l'entrainement de l'ensemble du modèle peut commencer. Nous n'allons procéder qu'à 6 sessions au lieu de 10 étant donné l'insuffisance de ressources dont nous disposons. Les chiffres relatifs aux différentes sessions sont repris dans la table 4.3. Nous remarquons qu'après les 6 sessions, le taux de précision de notre modèle est décent de l'ordre d'environ 58%.

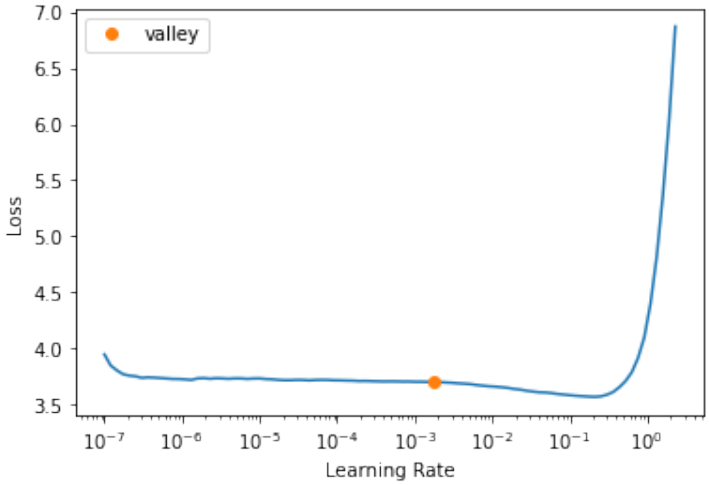


FIGURE 4.5 – Ajustement du langage au domaine : Taux d'apprentissage par rapport à la perte

epoch	train_loss	valid_loss	accuracy	perplexity	time
0	2.602870	2.471145	0.462722	11.835996	28:46

FIGURE 4.6 – Métriques du modèle d'ajustement après une session d'apprentissage.

Epoch	Training loss	Validation loss	Accuracy	Perplexity	Time
0	2.135286	2.282801	0.502813	9.804105	37 :13
1	2.160904	2.249899	0.508989	9.486776	37 :12
2	2.066479	2.147393	0.526786	8.562503	37 :12
3	1.990123	2.021648	0.545726	7.550758	37 :12
4	1.873042	1.890033	0.567577	6.619585	37 :12
5	1.912462	1.829383	0.579243	6.230042	37 :12

TABLE 4.3 – Métriques relatives à l'ajustement du modèle au domaine juridique

Malgré ce taux de précision, il nous semble qu'il soit possible d'augmenter celui-ci au delà de 60%. C'est à dire qu'à ce stade rendre notre modèle capable de prédire le mot suivant dans un texte de jurisprudence environ 2 fois sur trois.

6.2 Sauvegarde du modèle

Nous sauvegardons notre modèle pour une utilisation ultérieure. Exactement comme dans le cas du modèle de langage, nous devons sauvegarder le modèle ainsi que du vocabulaire correspondant est nécessaire.

La particularité à ce stade est la sauvegarde de l'encodeur [11] la représentation mathématique du modèle. En effet, jusqu'ici notre objectif était de déterminer le mot suivant. Lorsqu'il sera

11. L'utilité de l'encodeur est expliqué au point 5.3.3

question de travailler sur la tâche de classification nous dépouillerons le modèle de sa tâche de prédiction du mot suivant et remplacerons par celle dédiée à la prédiction de la recevabilité ou non de l'appel. Et pour ce fait nous utiliserons cet encodeur sauvegardé.

Il est aussi important de noter la possibilité d'ajouter de nouvelles connaissance au modèle généré à ce stade. Cela peut être nécessaire par exemple pour intégrer la connaissances de nouvelles décisions de justice. Il ne sera jamais nécessaire de recommencer tout depuis le début.

Le code utilisé pour procéder à l'entrainement de ce modèle est fourni dans l'annexe B.

7 Modèle de classification

Notre objectif était la création d'un modèle de classification capable de prédire la pertinence d'interjeter appel ou non d'une décision de justice d'une cour inférieure. Nous savons de quel format de donnée nous avons besoin comme abordé dans la section 4.3.2. Malheureusement nous n'avons pu constituer un jeu de données nous permettant d'atteindre notre objectif.

Cette impossibilité est principalement due à l'impossibilité de retrouver le numéro du dossier inférieur, pour une décision de justice prise en appel. La création d'un tel modèle est pour l'heure impossible dans le cadre belge.

8 Problèmes rencontrés

Dans le processus de réalisation de ces modèles, nous avons rencontré quelques problèmes qu'il serait important de prendre en compte pour la réalisation de ce genre de travail tout en disposant des ressources limités. Ces problèmes ont un lien étroit avec les outils dont nous disposons mais aussi la disponibilité ou non des données. Pour être concis par rapport à ceux-ci, nous allons les aborder séparément, commençant par ceux liés aux outils disponibles et ensuite la disponibilité et la qualité des données.

8.1 Disponibilité des outils

Pendant l'entrainement des différents modèles, nous avons fait face aux limites imposées par les différents outils disponibles. En effet, malgré l'utilisation d'un ordinateur assez puissant, nous nous sommes rendus compte que l'entrainement d'un modèle pouvait durer jusqu'à 10 jours.

Face à ce problème, nous avons cherché les outils permettant de réaliser un tel travail à moindre coût. C'est là que nous avons trouvé Colab (voir la section 3.2). Muni des données nécessairse pour entrainer notre modèle, nous avons vite déchanté. En effet, nous avons fait face à des interruptions qui impliquait de recommencer l'entrainement à zéro. Après plusieurs tentatives nous nous sommes mis à la recherche d'une alternative à Colab, et nous avons trouvé Kaggle (voir la section 3.3). Nous avons pu, grâce à ce service, procéder à la création du modèle de langage pré-entrainé qui plus tard a été utilisé dans la section 6.

Arrivé à la phase d'ajustement du modèle, nous avons une fois de plus fait face aux limites des services gratuits. En effet, chaque session dure plus d'une heure pour cette phase (1 heure 40 sur Colab et plus de deux heures sur Kaggle). Les instances Kaggle n'étant disponibles que pour une période de 12 heures, il nous a fallu faire un choix. D'une part nous devrions soit trouver une alternative à Kaggle pour finaliser cette phase ou alors réduire le nombre de sessions pour autant que nous continuons a avoir des résultats intéressants. Même en réduisant le nombre de

sessions il serait impossible de procéder à l'entrainement en un nombre suffisant de sessions et pouvoir télécharger les modèles créés.

En cherchant un peu plus, nous avons découvert Paperspace [12]. Moyenant un abonnement de 8 euros pour un mois, nous avons pu accéder au GPU ce qui a totalement permis d'accélérer notre entrainement. Passant de 2 heures sur Kaggle à 37 minutes.

8.2 Disponibilité et qualité des données

Nos tentatives d'en obtenir auprès des professionnels du droit n'ont pas été fructueuses. Et cette réticence à ce partage de donnée peut être compréhensible vu que généralement ce classement est parfois le fruit du travail de toute une carrière. Nous étions préparés à cette difficulté d'avoir accès à un jeu de donnée des décisions de justice, mais ne nous attendions pas à ce que les données trouvées soient impossibles à utiliser pour la tâche de classification.

Notre espoir était que les données mises à disposition des citoyens dans le cadre de l'Open Data devait permettre la transparence et celle-ci passe par la possibilité pour un citoyen de savoir quelles décisions de justice sont liées surtout quand il s'agit d'une décision dans une cour supérieure. Cette impossibilité est d'autant surprenante qu'avec la nouvelle identification européens de la jurisprudence (ECLI), il s'agit d'une action facilement réalisable lors de la mise en place de l'Open Data juridique.

L'absence d'un tel lien est un frein auquel il serait nécessaire de trouver une solution.

9 Conclusion

Notre objectif était de pouvoir en fin de ce chapitre proposer un modèle de classification totalement entrainé capable de prédire si l'appel d'une décision de justice en première instance sera recevable ou non en appel.

Nous avons dans un premier temps établi l'étendue du travail, procéder à la constitution du jeu de données nécessaire à l'entrainement et procédé à la création de deux modèles de langage, l'un ayant la capacité de comprendre le français et l'autre ayant une compréhension des textes juridiques.

Au cours de la recherche des données juridiques à utiliser pour notre tache de classification, nous avons fait face à l'impossibilité de lier les décisions de justice entre elles. Ce qui impliquait que notre projet initial était irréalisable.

Cependant, ce manque de lien entre les décisions, nous a permis d'identifier un problème auquel nous proposerons une solution dans le chapitre suivant.

12. Service permettant une utilisation gratuite ou à moindre coût du GPU. Accessible à l'adresse `https://www.paperspace.com/`

CHAPITRE 5

Open Data Case Law Representation

1 Introduction

Suite aux problèmes rencontrés dans la réalisation de notre modèle, dans ce chapitre, nous proposons une réflexion sur le formalisme idéal de représentation des décisions de justices publiées dans le cadre de l'Open Data.

Nous allons dans un premier temps présenter les motivations derrière notre proposition d'uniformisation. Ensuite nous ferons l'état des lieux en analysant la manière dont les documents juridiques sont publiées au sein de l'Union Européenne et pour finir nous présenterons l'*Open Data Case Law Representation (OD-CLR)*, regroupant nos recommandations dans l'optique de faciliter aux citoyens la consultations de ces documents.

2 Motivations

Lors de la réalisation de notre travail, et comme présenté dans la section 8.2 le plus grand problème que nous avons rencontré était lié à l'impossibilité de trouver le lien existant entre différentes décisions de justice. En ce sens où il était impossible, en Belgique tout du moins, de trouver par exemple pour une décision en appel le jugement du tribunal de première instance ayant été à l'origine de l'introduction de cet appel.

Le fait que cette information ne se trouve pas dans les méta-données des jugements publiés en open data aurait pu être contourné en utilisant d'autres techniques. Cependant en analysant les décisions des différentes cours d'appel, il n'est à aucun moment fait mention du numéro de dossier dans la cours inférieure de manière compréhensible. Cela nous a semblé déroutant étant donné la difficulté supplémentaire que cette absence de méta-donnée entraine pour les personnes consultant ces décisions. Elles sont dans l'incapacité de pouvoir remonter la chaine du litige jusqu'au début de son entrée dans le système judiciaire. Un peu comme si, cette donnée devait être gardée secrète. De plus, la plupart des décisions des tribunaux inférieurs ne sont tout bonnement pas publiées.

Ce constat nous a poussé à nous demander si c'était la pratique normale. Faisons cet exercice d'analyse pour tenter de répondre à cette question et voir quelle est la pratique dans d'autres pays.

3 État de lieux

Pour faire notre état des lieux, nous porterons notre attention sur la publication des décisions de justice au sein l'Union Européenne [1]. Nous allons tout d'abord commencer par discuter de l'identification ECLI [2] ensuite nous aborderons la qualité de données publiées.

3.1 Identification ECLI

Au niveau des décisions de justice, il faut reconnaître que l'existence d'une façon uniformisée d'identifier les décisions de justice au sein de l'Union Européenne est une bonne chose. Celle-ci permet de facilement savoir de quel pays provient une décision de justice, de quelle institution elle émane, l'année ainsi que le numéro de dossier au sein de l'institution ayant prise la décision. L'objectif de cette initiative telle que décrite sur le site BO ECLI [3] est de faciliter l'accès aux décisions de justice.

Cependant la normalisation de l'identification des documents est-elle suffisante ?

3.2 Données publiées

Une identification unifiée est une excellente initiative, mais semble ne pas être suffisante si les données ainsi publiées ne rencontrent pas le critères de qualité attendue et ne permettent pas au lecteur de pouvoir remonter la chaine du litige. Ce problème de qualité, ne peut être résolu que si en parallèle à l'identification unifiée, une norme définissant la manière de publier était définie.

Lors de la publication, il nous semble que certaines données sont tellement importante au même titre que l'identification du document que l'absence de celles-ci, sans impacter la qualité de la décision, impacte la qualité de sa publication.

Le rapport « *On-line Publication of Court Decisions in the EU* » [Mar+17] discute longuement de ce problème de qualité des décisions publiées. Il y a des disparités, et même entre les documents publiés au sein d'un même pays. Dans le chapitre consacré à la « *publication des décisions des cours* » [Mar+17, p. 11-18], nous pouvons voir à quel point l'accessibilité peut être mis à mal par l'absence de certaines méta-données.

Comme ECLI ne s'intéresse pas à la qualité, il nous a semblé judicieux de procéder à la définition d'une telle norme que nous appellons l'*Open data case law representation*. Celle-ci devrait décrire les méta-données qu'il faudrait avoir à minima pour considérer une publication comme étant de qualité.

4 L'OD-CLR

Pour répondre à ce problème de qualité, nous vous présentons l'OD-CLR, une spécification sur la façon de publier les décisions de justice. L'objectif de celle-ci est d'assurer la qualité de leur publication. L'avantage d'avoir une telle spécification est de pouvoir, sur base des points objectifs, comparer la qualité des données publiées par différentes institutions. De plus, cette représentation

1. Le Royaume-Uni compris
2. European Case Law Identification
3. Building on ECLI. Objectif de l'initiative dont le site web est accessible à l'adresse `https://bo-ecli.eu/bo-ecli/at-a-glance`

permettrait à toute personne intéressée d'avoir une idée des informations auxquelles s'attendre lors de la consultation des documents de justice publiés.

Nous avons identifié deux types de méta-données. Nous les classerons dans deux catégories, requises et optionnelles. Pour établir cette liste, nous avons pris en compte le contenu du rapport de BO-ECLI [Mar+17]. Il est à noter que notre expérience dans le développement du modèle présenté dans le chapitre 4, particulièrement celle liées aux recherches effectuées pour constituer le jeu de données nécessaire à notre travail, sera aussi largement utilisée.

Pour présenter ce modèle allons commencer par établir quelques définitions, celles-si seront suivies par la présentation des méta-données requises et optionnelles. Ensuite, nous parlerons des possibilité d'extension du modèle, des formats dans lesquels ces données devraient être accessibles, et pour finir nous aborderons le respect de la vie privée.

4.1 Définitions

Commençons par définir certains concepts clés qui seront utilisés dans notre proposition.

Date Tout au long de ce chapitre quand nous ferons référence à une date, ce sera la date telle que spécifiée dans le RFC3339 [KN02]

JSON Lorsqu'il sera fait mention du format json [4] il faudra l'entendre telle que spécifiée dans le RFC8259 [Bra17].

HTML5 Notre référence à ce format devra s'entendre au sens de la spécification « *HTML5* » [The+18] telle qu'elle a été approuvée par le W3C [5].

Anonymisation Nous parlerons d'anonymisation pour faire référence au concept du même mot qui est défini comme étant « *le processus par lequel l'information permettant l'identification est manipulé pour rendre l'identification des sujets quasi impossible* » [Esa15].

Pseudonymisation Contrairement à l'anonymisation, nous ferons référence à ce concept pour faire allusion « *au remplacement du nom ou des informations permettant une identification directe par les codes avec pour but que la données puisse être liée à une personne mais que celle-ci ne soit pas identifiable* » [Esa15].

4.2 Méta-données requises

Pour la publication des documents juridiques, nous avons identifiées les méta-données certaines comme étant utiles et nécessaires pour faciliter la consultation des documents publiées. Celles-ci sont l'ID, la langue, l'institution, la date de décision, le ID du document parent, le contenu et l'issue.

4.2.1 ID

Il s'agit de l'identification unique du document sous le format ECLI. Cette méta-donnée est importante pour faciliter la référence aux documents publiées au sein d'autres.

4. JavaScript Object Notation
5. World Wide Web Consortium

4.2.2 Langue

La plupart des jugements sont publiés dans une seule langue. Ce paramètre reste tout de même importante car elle permet par exemple de savoir quelle a été la langue originelle de publication du document. C'est celle-ci qui fait foi quant à la compréhension de l'essence de la décision. Cela est d'autant plus utile qu'il peut, dans certaines circonstances, être nécessaire de procéder à la traduction des documents juridiques de manière automatisée.

4.2.3 Institution

Cela va de soit, mais la publication de cette information est importante pour renforcer la transparence qui est l'un des objectifs derrière la publication des documents juridiques.

4.2.4 Date de la décision

Au fil du changement des lois, les décisions de justice pour un litige peuvent diverger. Fournir la date exacte de la prise de décision est important pour permettre de situer dans le temps.

4.2.5 Le contenu

Il va de soi que le contenu du document doit être fourni pour les jugements publié. Pour les documents publiés et dont le contenu n'a pas été fourni pour une raison donnée, le contenu ne peut pas être laissé vide mais devra être remplacé par cette raison.

4.2.6 Issue

Il est parfois difficile de savoir, pour les personnes n'étant pas des professionnels du droit, où trouver quelle a été l'issue d'un litige. Vu le nombre considérable de pages qui peuvent parfois constituer une décision et le caractère aléatoire du format de rédaction du contenu, la présence de cette méta-donnée est importante. Elle permettrait de palier à ce problème et au découragement que parfois peut entrainer la lecture d'un long jugement. En plus, cette information serait utile pour un justiciable qui chercherait des cas similaires au sien en fonction de l'issue.

4.2.7 ID du document parent

Cette information doit être fournie quand la décision publiée concerne un appel d'une décision prise précédemment par une autre institution. Par exemple, lorsque la décision publiée provient de la cours d'appel, le document parent auquel faire référence est la décision du tribunal de première instance. Cet identifiant doit, comme pour tout document publié, être au format ECLI.

4.3 Méta-données optionnelles

Certaines méta-données, que nous qualifierons d'optionnelles, ont aussi été identifiées. Leur publication accroit la qualité du document publié mais leur absence n'a aucun impact quant à la qualité minimale attendue. Il s'agit du résumé, des noms de juges, la jurisprudence et des textes légaux.

4.3.1 Résumé

Dans certains cas avoir un résumé du document publié peut être utile, surtout dans le cas où le contenu de la décision publiée est long. Ceci permet d'avoir les informations essentielles quant au contenu du document.

4.3.2 Noms de juges

Dans un objectif de transparence, il est important d'avoir la liste des juges ayant participé à la prise de décision. La fourniture de cette information permettrait de facilement trouver les différents dossiers sur lesquels un juge a été impliqué dans la prise de décision. Même si cette donnée, n'est pas important concernant la qualité de la publication, elle peut jouer un rôle important pour renforcer la confiance des citoyens dans le système.

4.3.3 Jurisprudence

Parfois, les décisions de justice sont prises et motivées par un ou plusieurs jugements précédents. Cette liste, si elle est fournie doit être composée des identifiants ECLI des documents de jurisprudence cités. Cette méta-donnée est complémentaire au résumé dans le sens il n'est plus nécessaire de parcourir la totalité du contenu pour les découvrir.

4.3.4 Textes légaux

Dans la motivation d'une décision de justice il est parfois fait référence aux textes légaux (par exemple : article de la constitution, article de lois, un décret, etc.). Dès lors fournir cette liste est une valeur ajoutée pour la qualité de la publication car elle permet à un citoyen consultant la page du document de savoir ce qui dans la loi a motivé le juge à prendre une décision et pas une autre. Tout comme pour la méta-donnée jurisprudence, et pour les même raison, celle-ci est complémentaire au résumé.

4.4 Extension de la représentation

Ces recommandations peuvent être étendues par une institution lors de sa publication en y ajoutant des méta-données supplémentaires qu'elle jugerait importantes et qui à ses yeux augmenteraient la qualité de ses publications.

4.5 Format de publication

Un autre point pour lequel il nous semble important d'apporter une recommandation technique, est le format de publication des documents juridiques. Il nous semble qu'il est important que le document publié soit au minimum consultable via le navigateur au format HTML5 pour faciliter la consultation depuis la plupart des terminaux actuels.

En plus de la publication dans ce format, nous recommandons qu'une version PDF du document soit publiée pour permettre la portabilité ainsi que la possibilité de le télécharger pour une utilisation hors ligne.

Dans la mesure du possible, chaque document devrait avoir sont équivalent au format json, où chaque méta-donnée serait définie comme un champ avec les valeurs adéquates.

4.6 Vie privée

Nous recommandons, que le contenu, telle que défini dans la section 4.2.5, soit dans tous les cas dépourvu de toute information à caractère personnel pouvant permettre l'identification des personnes privées impliquées dans le litige. Les techniques d'anonymisation ou de pseudonymisation peuvent être utilisées à cet effet.

Notre précision quant à l'implication des personnes privées impliquées dans les litiges, n'inclut pas les juges, les représentants des différents partis ou les fonctionnaires ayant à un moment donné ou un autre joué un rôle dans le processus ayant conduit à l'issue de la décision publiée. Il va de soi que les entreprises ne sont pas non plus à considérer étant donnée l'importance que cette information représente pour le public.

5 Conclusion

Dans ce chapitre, nous avons présenté l'OD-CLR, un ensemble de recommandations sur les méta-données qui doivent être présents lors de la publication des décisions de justice dans le cadre de l'Open Data.

Nous avons présenté quelles étaient nos motivations derrière ces recommandations et justifié chacune des méta-données que nous avons identifiées comme étant importantes pour les documents publiées. Nous avons aussi abordé le format de publication à utiliser pour rendre ces documents accessibles au plus grand nombre. De plus nous avons brièvement discuté de l'anonymisation ou la pseudonymisation de ces documents et défini les cas où ces techniques ne devraient pas être utilisées.

Ce chapitre nous permet de mettre un terme à ce travail. Mais qu'en est-il pour l'après ? Quelles sont les limites de notre travail ?

CHAPITRE 6

Apports, limites et perspectives

Au commencement de ce travail, il y avait une idée et un idéal que nous nous étions fixé comme objectif à atteindre. Mais comme tout travail, cette idée vit, évolue et peut même se transformer pour devenir autre chose. Dans ce chapitre nous allons parler de ce qu'apporte notre travail, des limites ainsi que les perspectives afin d'esquisser ce qui pourrait être fait dans la continuité de ce travail.

1 Apports

Ce travail nous a tout d'abord permis de constituer des jeux de données utilisables dans le cadre de l'entrainement d'un modèle de langage. Nous avons ainsi pu vérifier la possibilité d'entrainer un modèle de langage en partant de zéro ainsi que confirmer le fait qu'un tel entrainement ne nécessite pas trop de ressources. Le modèle de langage que nous avons ainsi crée peut être utilisé pour procéder à l'entrainement d'autre tâches en traitement du langage naturel faisant recours au transfert de connaissance.

Nous avons produit, au cours de ce travail, des scripts et notes permettant la reproduction des différentes étapes nécessaires à la création des différents modèles. Tout ceci est accessible à l'adresse `https://github.com/aieur/legal.ai`.

Les difficultés que nous avons rencontrées, particulièrement dans la phase de la constitution du jeu de données pour le modèle de classification, nous ont permis d'identifier le problème de qualité dont souffre la publication des décisions de justice. Ce problème est, nous semble-t-il, lié à un manque de standardisation sur les méta-données nécessaires pour considérer une publication comme étant de qualité. Nous avons donc proposé l'*OD-LCR*, une recommandation concernant les méta-données requises lors de la publication d'une décision de justice dans le cadre de l'open data.

2 Limites

Malgré les différentes vérifications et propositions faites dans la réalisation de ce travail, il est important de noter que notre objectif initial n'a pu être mené jusqu'au bout. Le modèle de classification que nous avions l'ambition de créer n'a pu être mené à bien. La qualité de la

publication des jurisprudences ne nous a pas permis de constituer un jeu de donnée pour son apprentissage.

N'ayant pas pu réaliser ce modèle de classification, dont la prétention n'était en aucun cas celui de pouvoir à terme remplacer un juge, nous n'avons pas non plus pu faire l'analyse du danger que peut représenter l'insertion d'une telle technologie dans le système juridique.

Cependant grâce à toutes ces barrières qui se sont posées sur notre chemin, nous voyons plusieurs possibilités devant nous et la continuité de ce travail. Mais que sont-elles ?

3 Perspectives

Ce travail, nous a permis de nous rendre compte du potentiel que présente le traitement du langage naturel, pas seulement dans le contexte juridique, mais bien au delà.

La constitution d'un ensemble de jeu de données permettant d'atteindre l'objectif initial de ce travail et la réalisation du modèle de classification disposant de la capacité de prédiction de la recevabilité de l'appel d'une décision de justice.

Ce travail, nous a aussi permis de nous rendre compte de la difficulté de trouver des modèles de langage pré-entrainés en français. Il serait intéressant de créer un espace de partage de modèles pré-entrainés utilisables et disposant d'un taux de précision supérieur à 40%. L'objectif serait, à terme, de disposer d'en avoir un pour chaque langue nationale au sein de l'Union Européenne.

La généralisation des recommandations faites dans le chapitre 5 à un cadre global pour aboutir à la proposition d'un draft RFC [1] semble être intéressant, car l'adoption d'une telle norme permettrait d'établir un standard de qualité quant à la publication de décision juridiques.

Il serait aussi important d'explorer la possibilité, en ayant recours à la méthode ULMFIT, d'augmenter le taux de précision d'un modèle pré-entrainé moyennant l'utilisation d'un nouveau jeu de données. Si cette possibilité était prouvée, cela permettrait de proposer une nouvelle approche dans l'entrainement d'un modèle de langage.

Nous pensons qu'il devrait être possible d'augmenter le taux de précision d'un modèle de langage en utilisant une combinaisons de plusieurs modèles différents. Arriver à prouver qu'une telle combinaison aurait un impact significatif permettrait de développer des modèles de langage plus efficaces.

Notre souhait est que la fin de ce travail ne soit pas la fin de l'exploration. Raison pour laquelle nous avons l'objectif de procéder à la création d'une librairie python au dessus de la librairie fastai. Celle-ci se focaliserait exclusivement sur le traitement du langage naturel. Ce librairie devrait avoir les fonctionnalités suivantes :

- Avoir la capacité d'utiliser des jeux de données partagés pour l'entrainement de modèles.
- Faciliter la création d'un modèle de langage pré-entrainé. Il suffirait de lui fournir le code ISO de la langue, et si celle-ci n'existe pas, il procéderait au téléchargement des données nécessaires à l'entrainement, le nettoyage automatique du jeu de données et la création du modèle.
- Détecter la puissance de calcul disponible et utiliser un jeu de données de taille adéquate.
- Se focaliser sur des tâches spécifiques du traitement du langage naturel, comme la classification et la reconnaissance d'entités.
- Être facilement intégrable dans les services via une api REST.

1. Request For Comment

Pour contribuer à notre proposition de l'OD-CLR, un projet qui aurait un impact significatif serait le développement d'un modèle de reconnaissance d'entités permettant l'extraction des articles de lois dans un texte. L'une des application de ce modèle serait de pouvoir faciliter l'anonymisation ou la pseudonymisation de décisions juridiques.

Il existe tellement de possibilités, qu'il serait impossible de toutes les énumérer. Nous nous sommes rendus compte lors de ce travail que l'imagination est la seule limite.

CHAPITRE 7

Conclusion

L'objectif de ce travail était de voir dans quelle mesure l'intelligence artificielle pourrait contribuer à rendre le processus juridique plus efficace.

Dans un premier temps, ce travail s'est intéressé à la justice pour identifier les objectifs de son informatisation. STEFAN les identifie comme étant, « *rendre les procédures efficaces, la transparence, la protection de données, la suppression de la corruption et rendre la gestion des ressources efficace* » [Ste10]. Une fois ces objectifs identifiés, l'intérêt s'est porté sur la justice prédictive qui s'avère ne pas être quelque chose de récent. En effet, prédire l'issue des dossiers, est un travail qui existe depuis aussi longtemps que l'est la jurisprudence. Cette dernière est utilisée par les avocats pour anticiper l'issue des affaires. Après ce constat, une analyse de la justice belge et son fonctionnement a été réalisée et a permis l'identification du point au sein du système juridique où la solution que nous comptions proposer avait tout son sens. Sur base des chiffres des affaires en attente, l'appel des jugements rendus en première instance est sorti du lot. L'objectif principal s'est tourné vers la réduction du nombre d'appels introduits auprès de la cour d'appel civile.

Cette identification, a tout naturellement suscité notre intérêt sur les aspects théoriques de l'intelligence artificielle. Dès lors, ses fondements et approches ont été explorés. Vu l'efficacité des réseaux de neurones et la multitude de possibilités qu'ils offrent, une section leur a été consacrée. Dans cette section il a été question de neurone, des fonctions d'activation, de rétro-propagation et des différentes architectures. Ces réseaux de neurones, devraient être entrainés pour comprendre la jurisprudence et permettre d'effectuer des tâches de classification.

Utiliser l'intelligence artificielle est une bonne chose, mais encore faut-il l'utiliser sur les décisions de justice. Celle-ci étant écrites en langage naturel, il était important de comprendre ce qu'était le langage naturel et les techniques permettant à la machine de les comprendre. Le langage naturel a ainsi été défini et l'importance du contexte est ressorti comme étant centrale dans sa compréhension. La machine devrait pouvoir avoir la connaissance des différents mots mais aussi comprendre leur utilisation en fonction du contexte, ensuite, la façon dont les mots pouvaient être représentés au sein d'un espace vectoriel qui n'en perdrait pas le contexte. Il était aussi important de comprendre quelles étaient les différentes approches dans le traitement du langage naturel et ses applications possibles.

L'objectif étant d'utiliser les réseaux de neurones, trois modèles de langage ayant recours au réseaux de neurones ont été identifiés, AWS-LSTM via la technique ULMFIT, BERT et GPT.

Tous les trois supportent le transfert de connaissance.

Pour le développement de la solution, le choix de la technique ULMFIT a semblé plus approprié car, comparé à BERT ou GPT, elle nécessite peu de ressources et permet d'atteindre les résultats quasi similaires. Dans le processus de développement, les différentes étapes ont été définies et les outils utilisés présentés. Il a ensuite été question de constituer les différents jeux de données. Face à l'impossibilité de trouver un modèle de langage pré-entrainé en français, un jeu de données pour créer un tel modèle a été fait sur base des articles Wikipédia. Au moment de réunir les décisions de justices nécessaires à la classification, nous nous sommes rendu compte de l'impossibilité de constituer ce jeu de donnée. La qualité dans la publication des décisions n'était pas au rendez vous.

L'entrainement du modèle de langage a été fait avec succès, suivi par celui de l'ajustement du langage à la jurisprudence. Par contre, l'absence de jeu de données pour la classification a rendu impossible la création du modèle final.

L'identification du problème dans la publication des décisions de justice, nous a conduit à la proposition d'une norme, l'OD-CLR définissant les méta-données qui devraient accompagner une décision de justice lors sa publication.

Arrivé à la fin de ce travail, il est évident qu'il pourrait être peaufiné par la constitution d'un jeu de données pour la classification ainsi qu'en procédant à des tests comparatifs avec l'utilisation d'autres modèles comme BERT et GPT. Il devrait aussi pouvoir être complété par l'analyse de l'impact en lien avec le biais pouvant provenir des données d'entrainement utilisées.

Acronymes

CBoW Continious bag of words
CRM Customer Relationship Management
ERP Enterprise Resource Planning
IDF Inverse document frequency
TFIDF Term Frequency - Inverse document frenquency
TF Term frequency

Bibliographie

[Hol97] Oliver Wendell Holmes. « The Path of the Law ». In : *Harward Law Review* 10 (1897).

[Tur50] Alan Turing. « Computing Machinery and Intelligence ». In : *Mind* LIX.236 (oct. 1950), p. 433-460. issn : 0026-4423. doi : 10.1093/mind/LIX.236.433. eprint : https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. url : https://doi.org/10.1093/mind/LIX.236.433.

[Cho56] Noam Chomsky. « Three models for the description of language ». In : *IRE Transactions on information theory* 2.3 (1956), p. 113-124.

[Cho57] Noam Chomsky. *Syntactic structures*. Mouton N.V, 1957.

[Fir57] John R Firth. « A synopsis of linguistic theory, 1930-1955 ». In : *Studies in linguistic analysis* (1957).

[Jon72] Karen Sparck Jones. « A statistical interpretation of term specificity and its application in retrieval ». In : *Journal of documentation* (1972).

[Bre+82] Joan Bresnan et al. « Cross-serial dependencies in Dutch ». In : *The formal complexity of natural language*. Springer, 1982, p. 286-319.

[FM83] Edward A Feigenbaum et Pamela McCorduck. *The fifth generation*. Addison-Wesley Pub., 1983.

[SM83] Gerard Salton et Michael J McGill. *Information Retrieval : an Introduction*. 1983.

[Hig84] James Higginbotham. « English is not a context-free language ». In : *The Formal Complexity of Natural Language*. Springer, 1984, p. 335-348.

[Shi85] Stuart M Shieber. « Evidence against the context-freeness of natural language ». In : *Philosophy, Language, and Artificial Intelligence*. Springer, 1985, p. 79-89.

[Elm90] Jeffrey L Elman. « Finding structure in time ». In : *Cognitive science* 14.2 (1990), p. 179-211.

[JSW90] Aravind K Joshi, K Vijay Shanker et David Weir. « The convergence of mildly context-sensitive grammar formalisms ». In : *Technical Reports (CIS)* (1990). [En ligne ; visité le 08-juin-2021], p. 539. url : https://repository.upenn.edu/cgi/viewcontent.cgi?article=1571%5C&context=cis_reports.

[Har91] Stevan Harnad. « Other bodies, other minds : A machine incarnation of an old philosophical problem ». In : *Minds and Machines* 1.1 (1991), p. 43-54.

[Bro+92] Peter F Brown et al. « Class-based n-gram models of natural language ». In : *Computational linguistics* 18.4 (1992), p. 467-480.

[WK92] Jonathan J Webster et Chunyu Kit. « Tokenization as the initial phase in NLP ». In : *COLING 1992 Volume 4 : The 14th International Conference on Computational Linguistics*. 1992.

[KI93] Bart Kosko et Satoru Isaka. « Fuzzy logic ». In : *Scientific American* 269.1 (1993), p. 76-81.

[WF94] Dekai Wu et Pascale Fung. « Improving Chinese tokenization with linguistic filters on statistical lexical acquisition ». In : *Fourth Conference on Applied Natural Language Processing*. 1994, p. 180-181.

[CYM95] S.Tamer Cavusgil, Poh-Lin Yeoh et Michel Mitri. « Selecting foreign distributors : An expert systems approach ». In : *Industrial Marketing Management* 24.4 (1995), p. 297-304. issn : 0019-8501. doi : https://doi.org/10.1016/0019-8501(95)00013-Z. url : https://www.sciencedirect.com/science/article/pii/001985019500013Z.

[JKS95] Ramesh Jain, Rangachar Kasturi et Brian G Schunck. *Machine vision*. T. 5. McGraw-hill New York, 1995.

[GS96] Ralph Grishman et Beth Sundheim. « Message Understanding Conference- 6 : A Brief History ». In : *COLING 1996 Volume 1 : The 16th International Conference on Computational Linguistics*. 1996. url : https://aclanthology.org/C96-1079.

[HS97] Sepp Hochreiter et Jürgen Schmidhuber. « Long short-term memory ». In : *Neural computation* 9.8 (1997), p. 1735-1780.

[LeC+98] Yann LeCun et al. « Gradient-based learning applied to document recognition ». In : *Proceedings of the IEEE* 86.11 (1998), p. 2278-2324.

[KN02] G. Klyne et C. Newman. *Date and Time on the Internet : Timestamps*. RFC 3339. RFC Editor, juill. 2002.

[Wan03] Sun-Chong Wang. « Artificial neural network ». In : *Interdisciplinary computing in java programming*. Springer, 2003, p. 81-100.

[SQ04] Wesley E Snyder et Hairong Qi. *Machine vision*. T. 1. Cambridge University Press, 2004.

[Abr05] Ajith Abraham. « Artificial neural networks ». In : *Handbook of measuring system design* (2005), p. 901-908.

[Mén+05] José Ramon Méndez et al. « Tokenising, stemming and stopword removal on anti-spam filtering domain ». In : *Conference of the Spanish Association for Artificial Intelligence*. Springer. 2005, p. 449-458.

[Sta05] Jason Stanley. *Knowledge and practical interests*. Clarendon Press, 2005.

[CL06] Herman Cappelen et Ernie Lepore. « Precis of Insensitive semantics ». In : *Philosophy and Phenomenological Research* 73.2 (2006), p. 425-434.

[Cas+07] Oscar Castillo et al. « Type-2 fuzzy logic : theory and applications ». In : *2007 IEEE international conference on granular computing (GRC 2007)*. IEEE. 2007, p. 145-145.

[CHH+09] Herman Cappelen, John Hawthorne, John P Hawthorne et al. *Relativism and monadic truth*. Oxford University Press, 2009.

[DS10] Ljiljana Dolamic et Jacques Savoy. « When stopword lists make the difference ». In : *Journal of the American Society for Information Science and Technology* 61.1 (2010), p. 200-203.

[Ste10] Ileana Stefan. « The Computerization Of The Judicial System ». In : *Curentul Juridic, The Juridical Current, Le Courant Juridique* 43 (déc. 2010), p. 163-167. url : `https://ideas.repec.org/a/pmu/cjurid/v43y2010p163-167.html`.

[SP10] J. Norvig Stuart et Russel Peter. *Artificial Intelligence, A modern approach*. Sous la dir. de Marcia J. Horton. 3rd. Prentice Hall Upper Saddle River, NJ, USA, 2010. isbn : 978-0-13-604259-4.

[VVV10] Tommi Vatanen, Jaakko J Väyrynen et Sami Virpioja. « Language Identification of Short Text Segments with N-gram Models. » In : *LREC*. 2010.

[GBB11] Xavier Glorot, Antoine Bordes et Yoshua Bengio. « Deep sparse rectifier neural networks ». In : *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop et Conference Proceedings. 2011, p. 315-323.

[Kum11] Ela Kumar. *Natural language processing*. IK International Pvt Ltd, 2011.

[NOC11] Prakash M Nadkarni, Lucila Ohno-Machado et Wendy W Chapman. « Natural language processing : an introduction ». In : *Journal of the American Medical Informatics Association* 18.5 (2011), p. 544-551.

[BL12] John A Bullinaria et Joseph P Levy. « Extracting semantic representations from word co-occurrence statistics : stop-lists, stemming, and SVD ». In : *Behavior research methods* 44.3 (2012), p. 890-907.

[GF12] Matthieu Guillaumin et Vittorio Ferrari. « Large-scale knowledge transfer for object localization in imagenet ». In : *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, p. 3202-3209.

[DL13] Tom Donaldson et Ernie Lepore. « 10 CONTEXT-SENSITIVITY ». In : *Routledge Companion to Philosophy of Language*. Routledge, 2013, p. 138-153.

[HM13] Hany Hassan et Arul Menezes. « Social text normalization using contextual graph random walks ». In : *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*. 2013, p. 1577-1586.

[MLS13] Tomas Mikolov, Quoc V Le et Ilya Sutskever. « Exploiting similarities among languages for machine translation ». In : *arXiv preprint arXiv :1309.4168* (2013).

[Mik+13] Tomas Mikolov et al. « Efficient estimation of word representations in vector space ». In : *arXiv preprint arXiv :1301.3781* (2013).

[Soc+13] Richard Socher et al. « Recursive deep models for semantic compositionality over a sentiment treebank ». In : *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, p. 1631-1642.

[Cho+14] Kyunghyun Cho et al. « Learning phrase representations using RNN encoder-decoder for statistical machine translation ». In : *arXiv preprint arXiv :1406.1078* (2014).

[Chu+14] Junyoung CHUNG et al. « Empirical evaluation of gated recurrent neural networks on sequence modeling ». In : *arXiv preprint arXiv :1412.3555* (2014).

[eC14] Conseil de l'Union EUROPÉENNE et Secrétariat général du CONSEIL. *Guide sur la visioconférence dans les procédures judiciaires transfrontières : : European e-justice*. Publications Office, 2014. DOI : doi/10.2860/81724.

[Kim14] Yoon KIM. « Convolutional Neural Networks for Sentence Classification ». In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar : Association for Computational Linguistics, oct. 2014, p. 1746-1751. DOI : 10.3115/v1/D14-1181. URL : https://aclanthology.org/D14-1181.

[PSM14] Jeffrey PENNINGTON, Richard SOCHER et Christopher D MANNING. « Glove : Global vectors for word representation ». In : *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, p. 1532-1543.

[Att15] Giusepppe ATTARDI. *WikiExtractor*. https://github.com/attardi/wikiextractor. 2015.

[Esa15] Samson ESAYAS. « The role of anonymisation and pseudonymisation under the EU data privacy rules : beyond the 'all or nothing'approach ». In : *European Journal of Law and Technology* 6.2 (2015).

[KF15] Andrej KARPATHY et Li FEI-FEI. « Deep visual-semantic alignments for generating image descriptions ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 3128-3137.

[KK15] Wołk KRZYSZTOF et Marasek KRZYSZTOF. « Neural-based Machine Translation for Medical Text Domain. Based on European Medicines Agency Leaflet Texts ». In : *Procedia Computer Science* 64 (2015). Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2015 October 7-9, 2015, p. 2-9. ISSN : 1877-0509. DOI : https://doi.org/10.1016/j.procs.2015.08.456. URL : https://www.sciencedirect.com/science/article/pii/S1877050915025910.

[LPM15] Minh-Thang LUONG, Hieu PHAM et Christopher D MANNING. « Effective approaches to attention-based neural machine translation ». In : *arXiv preprint arXiv :1508.04025* (2015).

[Tar15] DS TARASOV. « Natural language generation, paraphrasing and summarization of user reviews with recurrent neural networks ». In : *Materials of international conference" Dialog*. 2015.

[TE15] Enric TRILLAS et Luka ECIOLAZA. « Fuzzy logic ». In : *Springer International Publishing. DOI* 10 (2015), p. 978-3.

[Zhu+15] Yukun ZHU et al. « Aligning Books and Movies : Towards Story-Like Visual Explanations by Watching Movies and Reading Books ». In : *The IEEE International Conference on Computer Vision (ICCV)*. Déc. 2015.

[Bar16] Laura Barre. *Common law et droit continental : L'absence de culture juridique commune est-elle un mythe ?* [En ligne ; visité le 08-06-2022]. Juin 2016. URL : https://www.actu-juridique.fr/civil/common-law-et-droit-continental-labsence-de-culture-juridique-commune-est-elle-un-mythe.

[CAS16] Hans Christian, Mikhael Pramodana Agus et Derwin Suhartono. « Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF) ». In : *ComTech : Computer, Mathematics and Engineering Applications* 7.4 (2016), p. 285-294.

[Con+16] Alexis Conneau et al. « Very deep convolutional networks for text classification ». In : *arXiv preprint arXiv :1606.01781* (2016).

[Edw16] David Edward. « Le rôle de la jurisprudence dans le Common Law ». In : (2016).

[Inf16] RTL Info. *Tout le secteur de la justice Visé par les réductions de budget : Voici les chiffres qui leur font peur.* Mai 2016. URL : https://www.rtl.be/info/belgique/societe/tout-le-secteur-de-la-justice-vise-par-les-reductions-de-budget-voici-les-chiffres-qui-leur-font-peur-818921.aspx.

[Jou+16] Armand Joulin et al. « Fasttext. zip : Compressing text classification models ». In : *arXiv preprint arXiv :1612.03651* (2016).

[Lam+16] Guillaume Lample et al. « Neural Architectures for Named Entity Recognition ». In : *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*. San Diego, California : Association for Computational Linguistics, juin 2016, p. 260-270. DOI : 10.18653/v1/N16-1030. URL : https://aclanthology.org/N16-1030.

[LA16] Jeff Larson et Julia Angwin. *Machine bias.* Mai 2016. URL : https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[LJL16] Wootaek Lim, Daeyoung Jang et Taejin Lee. « Speech emotion recognition using convolutional and recurrent neural networks ». In : *2016 Asia-Pacific signal and information processing association annual summit and conference (APSIPA)*. IEEE. 2016, p. 1-4.

[ZL16] Xiang Zhang et Yann LeCun. « Text understanding from scratch ». In : *arXiv preprint arXiv :1502.01710* (2016).

[Age17] RTBF avec Agences. *L'action en justice aussi simple Qu'un achat en ligne ? La Chine Lance Le Tribunal Sur Internet.* Août 2017. URL : https://www.rtbf.be/article/l-action-en-justice-aussi-simple-qu-un-achat-en-ligne-la-chine-lance-le-tribunal-sur-internet-9686106.

[Bra17] T. Bray. *The JavaScript Object Notation (JSON) Data Interchange Format.* STD 90. RFC Editor, déc. 2017.

[HKS17] Dan Halperin, Lydia E Kavraki et Kiril Solovey. « Robotics ». In : *Handbook of discrete and computational geometry*. Chapman et Hall/CRC, 2017, p. 1343-1376.

[HAP17] Elnaz J Heravi, Hamed H Aghdam et Domenec Puig. « Classification of foods by transferring knowledge from ImageNet dataset ». In : *Ninth International Conference on Machine Vision (ICMV 2016)*. T. 10341. International Society for Optics et Photonics. 2017, p. 1034128.

[Li+17] Piji Li et al. « Deep recurrent generative decoder for abstractive text summarization ». In : *arXiv preprint arXiv :1708.00625* (2017).

[LP17] Kevin M Lynch et Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[Mar+17] van Opijnen Marc et al. *On-line Publication of Court Decisions in the EU*. BO-ECLI, 15 fév. 2017. URL : http://bo-ecli.eu/uploads/deliverables/Deliverable%20WS0-D1.pdf.

[Men+17] Afonso Menegola et al. « Knowledge transfer for melanoma screening with deep learning ». In : *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. IEEE. 2017, p. 297-300.

[MKS17] Stephen Merity, Nitish Shirish Keskar et Richard Socher. « Regularizing and optimizing LSTM language models ». In : *arXiv preprint arXiv :1708.02182* (2017).

[SS17] Sagar Sharma et Simone Sharma. « Activation functions in neural networks ». In : *Towards Data Science* 6.12 (2017), p. 310-316.

[Vas+17] Ashish Vaswani et al. « Attention is all you need ». In : *Advances in neural information processing systems*. 2017, p. 5998-6008.

[Dev+18] Jacob Devlin et al. « Bert : Pre-training of deep bidirectional transformers for language understanding ». In : *arXiv preprint arXiv :1810.04805* (2018).

[Fan18] Xuhui Fang. « Recent Development of Internet Courts in China ». In : *IJODR* 5 (2018), p. 49.

[HR18] Jeremy Howard et Sebastian Ruder. « Universal language model fine-tuning for text classification ». In : *arXiv preprint arXiv :1801.06146* (2018).

[Jul18] Balboni Julien. « Les TIC en chiffres ». In : *L'Echo* (2018). URL : https://www.lecho.be/economie-politique/belgique/federal/la-justice-se-modernise-avec-une-app-vieille-de-25-ans/10020618.html (visité le 09/06/2018).

[KR18] Taku Kudo et John Richardson. « SentencePiece : A simple and language independent subword tokenizer and detokenizer for Neural Text Processing ». In : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing : System Demonstrations*. Brussels, Belgium : Association for Computational Linguistics, nov. 2018, p. 66-71. DOI : 10.18653/v1/D18-2012. URL : https://aclanthology.org/D18-2012.

[NBK18] Irina Nikishina, Amir Bakarov et Andrey Kutuzov. « RusNLP : Semantic search engine for Russian NLP conference papers ». In : *International Conference on Analysis of Images, Social Networks and Texts*. Springer. 2018, p. 111-120.

[Rad+18] Alec Radford et al. « Improving language understanding by generative pre-training ». In : (2018).

[Sap18] Gilbert Saporta. *Une brève histoire de l'intelligence artificielle*. Conférence ENSAE-Alumni. Mars 2018. URL : https://hal-cnam.archives-ouvertes.fr/hal-02471743.

[Sen+18] Joeky T Senders et al. « An introduction and overview of machine learning in neurosurgical care ». In : *Acta neurochirurgica* 160.1 (2018), p. 29-38.

[SUW18] Carsten Steger, Markus Ulrich et Christian Wiedemann. *Machine vision algorithms and applications*. John Wiley & Sons, 2018.

[The+18] O'Connor Theresa et al. *HTML5*. W3C Recommendation. https ://www.w3.org/TR/2018/SPSD-html5-20180327/. W3C, mars 2018.

[Van18] Mark Van Hoecke. « Raisonnement juridique et droit comparé ». In : *Le droit malgré tout : Hommage à François Ost*. Sous la dir. de Jan Fagerberg, David C. Mowery et Richard R. Nelson. Brussels : Presses de l'Université Saint-Louis, 2018, p. 53-75.

[Wan+18] Alex Wang et al. « GLUE : A multi-task benchmark and analysis platform for natural language understanding ». In : *arXiv preprint arXiv :1804.07461* (2018).

[HK19] Michael Haenlein et Andreas Kaplan. « A brief history of artificial intelligence : On the past, present, and future of artificial intelligence ». In : *California management review* 61.4 (2019), p. 5-14.

[Lan+19] Zhenzhong Lan et al. « Albert : A lite bert for self-supervised learning of language representations ». In : *arXiv preprint arXiv :1909.11942* (2019).

[Liu+19] Yinhan Liu et al. « Roberta : A robustly optimized bert pretraining approach ». In : *arXiv preprint arXiv :1907.11692* (2019).

[Mar+19] Louis Martin et al. « Camembert : a tasty french language model ». In : *arXiv preprint arXiv :1911.03894* (2019).

[Par19] Ula La Paris. *Named Entity Recognition : la personnalisation de suggestions d'articles tech*. [En ligne ; Visité le 07-Juillet-2021]. Nov. 2019. url : `https://medium.com/data-by-saegus/ner-medium-articles-saegus-7ffec0f3188c`.

[Rad+19] Alec Radford et al. « Language models are unsupervised multitask learners ». In : *OpenAI blog* 1.8 (2019), p. 9.

[Bro+20] Tom B Brown et al. « Language models are few-shot learners ». In : *arXiv preprint arXiv :2005.14165* (2020).

[FZ20] Lorenzo Ferrone et Fabio Massimo Zanzotto. « Symbolic, Distributed, and Distributional Representations for Natural Language Processing in the Era of Deep Learning : A Survey ». In : *Frontiers in Robotics and AI* 6 (2020), p. 153.

[HG20] J. Howard et S. Gugger. *Deep Learning for Coders with Fastai and Pytorch : AI Applications Without a PhD*. O'Reilly Media, Incorporated, 2020. isbn : 9781492045526. url : `https://books.google.no/books?id=xd6LxgEACAAJ`.

[Jea20] Jacquet Jean-Marie. « Techniques d'intelligence artificielle ». In : Université de Namur, déc. 2020.

[Joh20] Vandenberghe Johan. « La justice prédictive : Application au droit des marques ». Mém. de mast. Namur, BE : Universite de Namur, 2020.

[LV20] Stefan Leijnen et Fjodor van Veen. « The neural network zoo ». In : *Multidisciplinary Digital Publishing Institute Proceedings*. T. 47. 1. 2020, p. 9.

[LLS20] Zhiyuan Liu, Yankai Lin et Maosong Sun. *Representation Learning for Natural Language Processing*. Springer Nature, 2020.

[Mar+20] Louis Martin et al. « CamemBERT : a Tasty French Language Model ». In : *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online : Association for Computational Linguistics, juill. 2020, p. 7203-7219. URL : https://www.aclweb.org/anthology/2020.acl-main.645.

[Par+20] Kyubyong Park et al. « An Empirical Study of Tokenization Strategies for Various Korean NLP Tasks ». In : *arXiv preprint arXiv :2010.02534* (2020).

[Wik20] Wikipedia Contributors. *Étiquetage morpho-syntaxique — Wikipedia*. [En ligne ; Visité le 07-Juilet-2021]. 2020. URL : https://fr.wikipedia.org/w/index.php?title=%C3%89tiquetage_morpho-syntaxique%5C&oldid=172456303.

[WP20] Adrian de Wynter et Daniel J Perry. « Optimal subarchitecture extraction for bert ». In : *arXiv preprint arXiv :2010.10499* (2020).

[Col21] Collège des cours et tribunaux. *Les statistiques annuelles des cours et tribunaux. Données 2012-2020*. [En ligne, Visité le 26 juin 2022]. Brussels, BE, 2021. URL : https://www.rechtbanken-tribunaux.be/sites/default/files/college/stat/justice-en-chiffres_2012-2020.pdf.

[Cou21] Cour de justice de l'Union européenne. *Affaire C-301/20 — ARRÊT DE LA COUR (sixième chambre)*. [En ligne ; Visité le 6-Juillet-2021]. 2021. URL : https://eur-lex.europa.eu/legal-content/FR/TXT/HTML/?uri=CELEX:62020CJ0301%5C&qid=1625658465291%5C&from=FR.

[Dan21] Otri Dania. « Artificial Intelligence and Predictive Justice ». Mém. de mast. The address of the publisher : Universite de Namur, jan. 2021.

[Jal+21] Amir Jalilifard et al. « Semantic sensitive TF-IDF to determine word relevance in documents ». In : *Advances in Computing and Network Communications*. Springer, 2021, p. 327-337.

[Lar21a] Larousse. *Intelligence artificielle*. 2021. URL : https://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257 (visité le 31/05/2021).

[Lar21b] Larousse. *Justice*. 2021. URL : https://www.larousse.fr/dictionnaires/francais/justice/45236 (visité le 14/11/2021).

[Lar21c] Éditions Larousse. *Définitions : informatiser*. https://www.larousse.fr/dictionnaires/francais/informatiser/43001. [Online ; accessed 16-May-2021]. 2021.

[Mer21] Annick Merckx. *Digitalisation de la Justice : le ministre veut relancer le processus, le chantier est immense*. [En ligne, Visité le 24 juin 2022]. Avr. 2021. URL : https://www.rtbf.be/article/digitalisation-de-la-justice-le-ministre-veut-relancer-le-processus-le-chantier-est-immense-10752519.

[WK21] Ben Wang et Aran Komatsuzaki. *GPT-J-6B : A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax. Mai 2021.

[Wik21a] Wikipédia. *Machine learning — Wikipedia, The Free Encyclopedia*. [En ligne ; visité le 04-juin-2021]. 2021. URL : https://en.wikipedia.org/wiki/Machine_learning.

[Wik21b] Wikipedia Contributors. *File :Hierarchie chomsky.svg — Wikipedia*. [En ligne ; Visité le 04-Juin-2021]. 2021. URL : https://commons.wikimedia.org/wiki/File:Hierarchie_chomsky.svg.

[Wik21c] Wikipedia Contributors. *File :Typical cnn fr.png— Wikipedia*. [En ligne ; Visité le 02-Novembre-2021]. 2021. URL : https://commons.wikimedia.org/wiki/File:Typical_cnn.png.

[Wik21d] Wikipedia Contributors. *Softmax function — Wikipedia*. [En ligne ; Visité le 01-Juilet-2021]. 2021. URL : https://en.wikipedia.org/w/index.php?title=Softmax_function%5C&oldid=1030172347.

[Wik21e] Wikipedia Contributors. *tf–idf — Wikipedia*. [En ligne ; Visité le 20-Juin-2021]. 2021. URL : https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf%5C&oldid=1023875568.

[Bel22] Chambre des réprésentants de Belgique. *Le pouvoir judiciaire - Division du droit*. 2022. URL : https://www.dekamer.be/kvvcr/pdf_sections/pri/fiche/fr_21_00.pdf (visité le 10/06/2022).

[Eur22] Eurostat. *Ménages - niveau d'accès à l'internet*. https://ec.europa.eu/eurostat/databrowser/view/isoc_ci_in_h. [En ligne, Visité le 01 avril 2022]. 2022.

[Ins22] Justice en ligne Institut d'Études sur la Justice. *Lexique*. 2022. URL : https://www.justice-en-ligne.be/-Lexique-?id_article=125%206 (visité le 10/06/2022).

[Lar22] Larousse. *Prédire*. 2022. URL : https://www.larousse.fr/dictionnaires/francais/pr%C3%A9dire/63424 (visité le 17/06/2022).

[Ser22] Braudo Serge. *Litige - Définition*. 2022. URL : https://www.dictionnaire-juridique.com/definition/litige.php (visité le 10/06/2022).

[Eco] SPF Economie. *Les TIC en chiffres*. URL : https://economie.fgov.be/fr/themes/line/les-tic-en-belgique/les-tic-en-chiffres (visité le 01/12/2021).

[fra] Académie française. *La 9e édition*. [En ligne ; visité le 10-juin-2021]. URL : https://www.academie-francaise.fr/le-dictionnaire/la-9e-edition.

ANNEXE A

Code de l'entrainement du modèle de langage

```
# Installation des d\'{e}pendances
!pip install fastai

# Import des librairie
from fastai.text.all import *
import os
import tarfile

# D\'{e}finition des variables
path = Path(os.getcwd())
data_set = 'wiki-28k'
data_dir = path / 'data'
model_path = data_dir / 'models'
set_path = data_dir / 'sets'
set_path.mkdir(parents=True, exist_ok=True)
model_path.mkdir(parents=True, exist_ok=True)
uncompressed_dataset = path / data_set

# R\'{e}cup\'{e}ration des datasets si applicable
if not data_dir.exists():
  !wget https://www.comwes.eu/data.tar.gz
  file = tarfile.open('data.tar.gz')
  file.extractall()
  file.close()

if not uncompressed_dataset.exists():
  file = tarfile.open('{0}/{1}'.format(set_path,f'{data_set}.tar.gz'))
  file.extractall()
  file.close()

lang = 'fr'
sentence_piece_model = model_path /'spm'/'spm.model'

# D\'{e}finition du tokeniseur \`{a} utiliser
```

```
# pouvons aussi utiliser WordTokenizer(lang=lang)
tok = SentencePieceTokenizer(lang=lang, sp_model=sentence_piece_model)

# D\'{e}finition du jeu de donn\'{e}e. la tokenisation se fait ici
data_blk = DataBlock(
  blocks=TextBlock.from_folder(
    uncompressed_dataset,
    is_lm=True,
    tok=tok
  ),
  get_items=get_files,
  splitter=RandomSplitter(valid_pct=0.1, seed=42)
)

dataloader = data_blk.dataloaders(
  uncompressed_dataset,
  path=uncompressed_dataset,
  verbose=True,
  num_workers=18
)
len(dls.train), len(dls.valid)

dataloader.show_batch(max_n=1)

# D\'{e}finition du mod\`{e}le d'entrainement
learner = language_model_learner(
  dls, AWD_LSTM,
  drop_mult=0.5,
  pretrained=False,
  metrics=[accuracy, Perplexity()]
)

# D\'{e}couverte de taux d'apprentissage \`{a} utiliser
learner.lr_find()

# Entrainnement du mod\'{e}le
from fastai.callback.progress import CSVLogger
learner.path = model_path.absolute()
learning_rate = 1e-2
number_epoch = 10
cbs = [CSVLogger(fname=f'history.csv')]
learner.unfreeze()
learner.fit_one_cycle(number_epoch, learning_rate/2, cbs=cbs)

# Sauvegarde du mod\'{e}le
langage_model_files = [path / f'{lang}_wiki-28k', path / f'{lang}_wiki-28k_vocab.pkl']
learner.to_fp32().save(langage_model_files[0].absolute(), with_opt=False)
with open(langage_model_files[1], 'wb') as f:
  pickle.dump(learner.dls.vocab, f)

```

ANNEXE B

Code utilisé pour l'ajustement du modèle de langage au domaine

```
# Installation des d\'{e}pendances
!pip install fastai

# Import des librairie
from fastai.text.all import *
import os
import tarfile

# D\'{e}finition des variables
path = Path(os.getcwd())
lang='fr'
data_set = 'juris'
data_dir = path / 'data'
model_path = data_dir / 'models'
set_path = data_dir / 'sets'
set_path.mkdir(parents=True, exist_ok=True)
model_path.mkdir(parents=True, exist_ok=True)
langage_model_files = [(model_path / f'{lang}_wiki28k_model').absolute(), (model_path / f'{
  lang}_wiki28k_vocab').absolute()]


# R\'{e}cup\'{e}ration des datasets si applicable
if not data_dir.exists():
  !wget https://www.comwes.eu/{data_set}.tar.gz
  file = tarfile.open('{}.tar.gz'.format(data_set))
  file.extractall()
  file.close()

if not uncompressed_dataset.exists():
  file = tarfile.open('{0}/{1}'.format(set_path,f'{data_set}.tar.gz'))
  file.extractall()
  file.close()

sentence_piece_model = model_path /'spm'/'spm.model'
```

```
34
35  # D\'{e}finition du tokeniseur \`{a} utiliser
36  # pouvons aussi utiliser WordTokenizer(lang=lang)
37  tok = SentencePieceTokenizer(lang=lang, sp_model=sentence_piece_model)
38
39  # D\'{e}finition du jeu de donn\'{e}e. la tokenisation se fait ici
40  data_blk = DataBlock(
41    blocks=TextBlock.from_folder(
42      uncompressed_dataset,
43      is_lm=True,
44      tok=tok
45    ),
46    get_items=get_files,
47    splitter=RandomSplitter(valid_pct=0.1, seed=42)
48  )
49
50  dls = data_blk.dataloaders(
51    uncompressed_dataset,
52    path=uncompressed_dataset,
53    verbose=True,
54    num_workers=18
55  )
56  len(dls.train), len(dls.valid)
57
58  dataloader.show_batch(max_n=1)
59
60  # D\'{e}finition du mod\`{e}le d'entrainement
61  learner = language_model_learner(
62    dls, AWD_LSTM,
63    drop_mult=0.5,
64    pretrained=True,
65    pretrained_fnames=langage_model_files,
66    metrics=[accuracy, Perplexity()]
67  )
68
69  # D\'{e}couverte de taux d'apprentissage \`{a} utiliser
70  learner.lr_find()
71
72  # Entrainement de la derni\`{e}re couche
73  learning_rate = 1e-2
74  learner.fit_one_cycle(1, learning_rate)
75
76  # Entrainnement de toutes les couches
77  from fastai.callback.progress import CSVLogger
78  learner.path = model_path.absolute()
79  number_epoch = 6
80  cbs = [CSVLogger(fname=f'history.csv')]
81  learner.unfreeze()
82  learner.fit_one_cycle(number_epoch, learning_rate, cbs=cbs)
83
84  # Sauvegarde du mod\'{e}le
85  langage_fine_tuned_files = [(path/f'{data_set}_ft_model').absolute(),
86  (path/f'{data_set}_ft_vocab.pkl').absolute()]
87
88  learner.to_fp32().save(langage_fine_tuned_files[0], with_opt=False)
89  learner.save_encoder((path/f'{data_set}_ft_encoder').absolute())
90  with open(langage_fine_tuned_files[1], 'wb') as f:
```

```
91      pickle.dump(learner.dls.vocab, f)
92
```