Analyse syntaxique automatique du pidgin-créole du Nigeria à l'aide d'un transformer (BERT) : Méthodes et Résultats

Kirian Guiller

Mémoire de Recherche

ILPGA
Université Sorbonne Nouvelle
Paris III
France

Encadrant : Kim Gerdes
Co-encadrant : Sylvain Kahane

1er Septembre 2020

Attestation de non-plagiat

Je, soussigné Kirian GUILLER, déclare avoir rédigé ce travail sans aides extérieures ni sources autres que celles qui sont citées. Toutes les utilisations de textes préexistants, publiés ou non, y compris en version électronique, sont signalées comme telles. Ce travail n'a été soumis à aucun autre jury d'examen sous une forme identique ou similaire, que ce soit en France ou à l'étranger, à l'université ou dans une autre institution, par moi-même ou par autrui.

1 septembre 2020

Remerciements

Je tiens à remercier toutes les personnes qui ont aidé à la rédaction de ce mémoire. Mes remerciements s'adressent en particulier à Kim Gerdes et Sylvain Kahane qui ont encadré mon travail et m'ont orienté dans une directions qui a su éveiller et développer ma curiosité scientifique. Je tiens aussi à remercier l'équipe ALMAnaCH d'INRIA Paris, dirigée par Benoît Sagot, pour l'accueil et l'infrastructure que j'ai reçu. Les modèles présentés dans ce mémoire ont été entraînés sur les serveurs de l'INRIA. De plus, je voudrais remercier le Labex EFL "Empirical Foundations of Linguistics", piloté par Christel Préterre, pour m'avoir fourni une bourse de recherche. Je remercie aussi Bernard Caron, chef du projet ANR NaijaSynCor, ainsi que les autres membres du projet, pour l'aide apportée sur certaines de mes problèmatiques ainsi que leurs travaux sur le NaijaSyncor Treebank. Je souhaite aussi remercier par avance les membres du jury, Iris Eshkol-Taravella, Rachel Bawden et Bernard Caron pour la relecture de mon mémoire ainsi que leurs participations à la soutenance. Enfin, je tiens à remercier Marine Courtin, YiXuan Li, SoYoung Park, Gaël Guibon, Emmett Strickland, Benjamin Muller, Pierre Rochet, Pedro J. Ortiz et bien d'autres collègues pour les échanges, professionnels ou non, que nous avons pu partager.

Résumé

Ce mémoire a pour but de retracer le travail que nous avons effectué pour la réalisation d'un analyseur de dépendances syntaxiques pour le naija, le pidgin-créole de l'anglais parlé au Nigeria, afin de montrer l'influence du transfert de connaissances depuis l'anglais pour compenser le peu de ressources brutes disponibles en naija. Nous avons entraîné sur des treebanks annotés en syntaxe un transformer BERT couplé à une architecture d'attention bi-affine. Notre travail s'inscrit dans le projet ANR NaijaSynCor (en partenariat avec les laboratoires du Llacan et de Modyco) dans le cadre duquel un corpus oral de 500 000 mots a été réalisé dont un quart a été annoté en syntaxe. Notre participation au projet a consisté d'une part à fournir aux linguistes une pré-annotation syntaxique afin de les aider dans leurs travaux d'annotation et d'autre part à enrichir le treebank en annotant automatiquement sans correction manuelle 350 000 mots. Nous proposons dans ce mémoire de répondre à la question suivante : Comment développer un analyseur pour le naija en s'aidant de sa proximité avec l'anglais pour lequel il existe un grand nombre de ressources ?

Contents

Chapitre 1

Introduction

Le naija est une langue du Nigeria parlée par plus de 100 millions de locuteurs. Cette langue, de par ses origines linguistiques, est proche de l'anglais et ses normes sont encore en cours d'établissement. À l'origine un simple pidgin de l'anglais, le naija s'est ensuite développé à Lagos (ville la plus peuplée d'Afrique avec 25 millions d'habitants) comme langue autonome [Deuber, 2005].

Améliorer les outils et ressources de cette langue pourrait permettre aux linguistes d'étudier l'émergence d'un pidgin-créole se nationalisant afin d'améliorer les théories existantes du développement de créoles et de langues de contact. C'est dans cette optique que le projet ANR NaijaSynCor est mené en partenariat avec les laboratoires du Llacan, de Moydco et de l'Université d'Ibadan. L'objectif du projet NSC est d'analyser cette langue encore très peu étudiée en produisant un corpus oral avec des points d'enquête répartis sur l'ensemble du territoire afin de faire une analyse sociolinguistique.

Notre participation dans ce projet consiste en la création d'un analyseur syntaxique qui tire parti des marqueurs prosodiques pour construire des arbres de dépendances de l'énoncé. Le naija est une langue qui ne comporte pas de corpus brut conséquent (comme un Wikipédia ou un grand nombre de sites web aspirables) ce qui représente une contrainte majeure qu'il nous faut contourner. Pour se faire, nous allons nous inspirer des travaux existants en analyse syntaxique ainsi qu'en apprentissage par transfert et déterminer les facteurs importants de réussite lors de la réalisation d'analyseurs syntaxiques pour des pidgins et créoles. Le but étant de réaliser l'analyseur du naija le plus performant possible avec un nombre de ressources limité.

L'analyseur obtenu a deux objectifs. L'un est d'aider les annotateurs du projet NaijaSynCor en leur proposant une pré-annotation du corpus, l'autre est d'enrichir artificiellement le corpus avec des énoncés annotés sans révision humaine.

Chapitre 2

Contexte Scientifique

Le cadre de ce mémoire n'est pas limité à un seul domaine scientifique et se situe à la jonction entre la linguistique, l'informatique et les mathématiques. Plus précisément, nous utilisons des techniques d'apprentissage automatique pour créer des modélisations syntaxiques du naija. Cette partie II du mémoire permet de donner les notions essentielles à la bonne compréhension des travaux et résultats ici présents ainsi que de contextualiser notre contribution par rapport à l'état de l'art actuel.

2.1 Notions clés

2.1.1 Notions typologiques

Pour bien comprendre nos explications sur le naija faites en 2.2, nous souhaitons donner les définitions que nous utilisons pour classifier les langues de contact.

Pidgin : Un pidgin est une langue issue d'un contact entre au moins deux langues généralement dans un contexte d'échange. Par définition, le pidgin n'est la langue native d'aucun individu (Bakker 2008).

Pidgin-créole : C'est un pidgin qui est devenu la langue principale d'un groupe d'individu et/ou la langue native de certains de ses locuteurs (Bakker 2008).

Créole : C'est un pidgin-créole qui s'est élargi et est devenu la langue principale d'une ethnie ou une communauté entière. (Bakker 2008).

D'après les définitions de (Bakker 2008) que nous employons, le critère de différenciation entre pidgin, pidgin-créole et créole se situe au niveau de l'appropriation de cette langue par ses locuteurs et non pas au niveau de sa structure. En conséquences des critères sociaux mentionnés ci-dessus, les créoles et pidgins-créoles sont bien souvent plus sophistiqués linguistiquement que

leurs pidgins respectifs.

2.1.2 Notions linguistiques

Les analyses du naija que nous produisons sont des analyses en grammaire de dépendance. Nous rappelons ici ce que nous entendons par grammaire de dépendance ainsi que des notions importantes de ce champ scientifique.

Grammaire de dépendance (DG) : Cette approche de la grammaire consiste à décrire les connexions qui ont lieu entre les mots d'un énoncé. Chaque connexion dispose d'un gouverneur et d'un dépendant ce qui rend donc cette connexion asymétrique. Le dépendant est généralement un argument ou un modifieur du gouverneur. Pour une explication plus détaillée de la grammaire de dépendance, il peut être utile de se référer aux travaux de Sylvain Kahane qui définissent la grammaire de dépendance (Kahane 2001). De plus, le livre (Kahane and Gerdes 2020) constitue aussi une très bonne ressource pour quiconque souhaitant approfondir ses connaissances de la DG.

Arbre de dépendance : L'application de cette grammaire à un énoncé complet (par exemple une phrase) nous permet d'obtenir un arbre de dépendances. Cet arbre peut être modélisé par un graphe où chaque nœud du graphe représente une unité syntaxique et chaque arête dirigée du graphe représente une connexion syntaxique entre deux unités. Pour être en accord avec les principes de la syntaxe de dépendance, un arbre syntaxique est un graphe qui possède les propriétés suivantes :

- Le graphe possède un unique nœud racine qui ne possède aucun gouverneur.
- Tout nœud du graphe, excepté la racine, possède un unique gouverneur.
- Le graphe est acyclique.

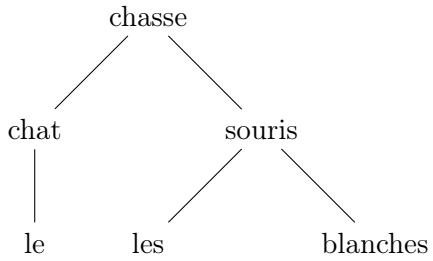

Figure 2.1: Arbre de dépendance syntaxique non-linéarisé et pour l'énoncé *le chat chasse les souris blanches*. On observe une racine unique *chasse* possédant deux dépendants.

Token : La définition standard de token désigne une séquence de caractères contigus située entres deux caractères de séparation (espace, tabulation, retour à la ligne) ou ponctuation. Par exemple, l'unité syntaxique "abat-jour' est constitué de deux tokens. Par souci de simplicité,

nous utilisons lors de nos travaux une définition légèrement modifiée : Le token est le terme informatique utilisé pour désigner les nœuds de l'arbre. Il est donc l'équivalent du terme linguistique *unité syntaxique* et du terme mathématique *nœud*.

Arc (syntaxique) : Terme utilisé pour désigner une arête de l'arbre. L'arc est toujours dirigé. Cette direction dépend de la convention utilisée. Dans ce mémoire, lorsque nous modélisons schématiquement des arcs syntaxiques, nous représenterons une flèche dont la racine représente le gouverneur et la pointe le dépendant.

Relation/fonction/label (syntaxique) : Terme utilisé pour désigner l'étiquette d'une connexion syntaxique. Le principe général est que deux connexions qui ont la même étiquette relationnelle ont des propriétés communes, tandis que les connexions ayant des étiquettes distinctes ont des propriétés distinctives. Les propriétés en question varient selon les cadres théoriques et peuvent concerner les seules propriétés de la connexion étudiée (comme en UD, voir 2.3.2) ou l'ensemble du paradigme de commutation du dépendant (comme en SUD, voir 2.3.3). La relation peut prendre une valeur discrète contenue dans une liste d'étiquettes possibles définies par le système d'annotation du treebank (voir définition en 2.3). Les termes "relation" et "fonction" syntaxique sont plus couramment utilisés en linguistique. Le terme "label" syntaxique est lui plus souvent utilisé en informatique. Il nous arrivera donc de changer le terme utilisé en fonction du contexte.

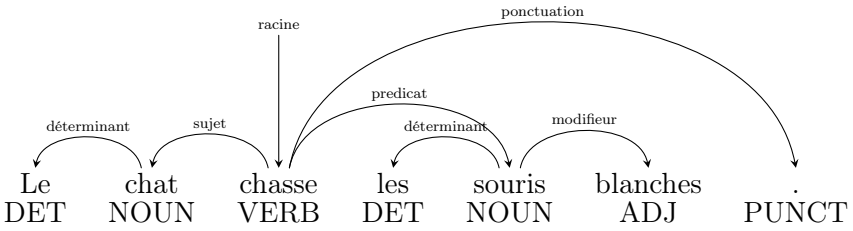

Figure 2.2: Arbre de dépendance syntaxique linéarisé et avec ajout d'étiquettes morpho-syntaxiques et des fontions syntaxiques pour l'énoncé *le chat chasse les souris blanches*

L'ancre de l'arbre syntaxique : L'élément de la phrase le plus haut dans l'arbre syntaxique, qui gouverne tous les autres mots, est traditionnellement appelé en linguistique la racine de la phrase. Sur la figure 2.2, cette racine est le token *chasse* et l'on peut voir qu'un arc entrant *racine* pointe sur ce token.
Par souci de modélisation en informatique, il est de coutume de rajouter au sommet de la structure mathématique un nœud fictif afin de pouvoir modéliser un arc entrant (unique) qui pointe vers la racine syntaxique de la phrase. Il se trouve que ce concept est généralement désigné par *root* (terme anglais pour désigner *racine*) et cela peut porter confusion avec l'emploi linguistique.
Pour éviter toute ambiguïté, nous utiliserons le terme ancre pour désigner cette *root* fictive informatique et le terme *tête de la phrase* pour désigner le token qui gouverne tous les autres tokens de la phrase. Nous pouvons voir sur la figure 2.3 un exemple d'un tel ajout de nœud fictif

en haut de l'arbre.

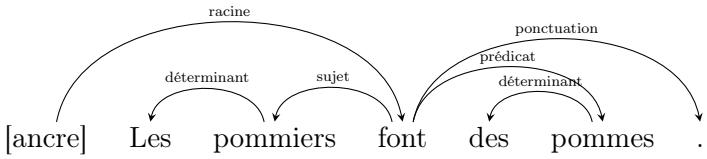

Figure 2.3: Arbre syntaxique linéarisé avec ajout d'une ancre pour l'énoncé *Les pommiers font des pommes.*

Projectivité d'un arbre : Un arbre syntaxique est projectif lorsque ses arcs ne se croisent pas entre eux. Inversement, un arbre est non projectif quand au moins deux de ses arcs se croisent. La présence et la proportion de construction projective dans une langue dépend de la langue en question ainsi que du formalisme choisi de syntaxe de dépendance.

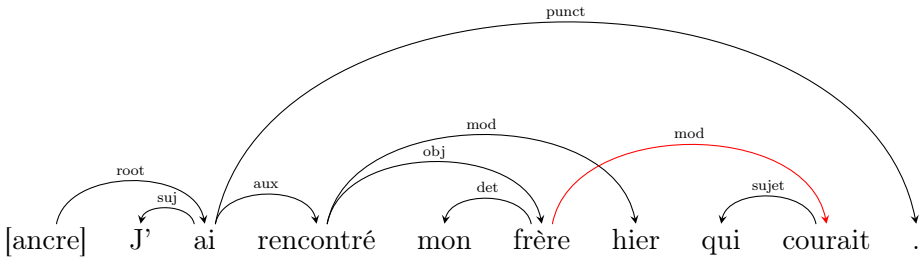

Figure 2.4: Arbre syntaxique linéarisé d'un arbre non projectif pour l'énoncé *J'ai rencontré mon frère hier qui courait.*

Partie du discours (Part Of Speech/POS) : Les parties du discours ont originellement été créées dans le but de classifier les mots en fonction de leurs distributions morpho-syntaxiques. Le but étant de regrouper dans de mêmes classes des mots qui se comportent de manières similaires. Par exemple, en français, la classe des adjectifs regroupe tous les mots qui sont utilisés apporter une information descriptive sur les noms. Le nombre de parties du discours est arbitraire puisqu'en théorie, il pourrait en exister autant qu'il existe de mots (puisque tous les mots sont différents). Un compromis est généralement trouvé dans chaque langue pour avoir un nombre de catégories assez élevé pour permettre la catégorisation des mots et assez bas pour permettre aux locuteurs de les mémoriser. En français, on a longtemps possédé une grammaire avec neuf parties du discours héritées du latin. Cela comportait des contradictions puisque le français est un descendant du latin et possède des déterminants que le latin n'a pas.

Analyseur syntaxique (parseur) : L'analyseur syntaxique désigne un programme informatique qui analyse la structure syntaxique d'un énoncé. Dans le domaine scientifique et informatique, il est plus courant d'utiliser le terme "parseur" (anglicisme du terme "parser", lui-même dérivé de l'ancien français. En anglais originalement le terme était plutôt utilisé pour désigner l'annotation (manuelle ou automatique) des parties du discours. Aujourd'hui, le pars-

ing est plutôt distingué du tagging, le premier désigne l'attribution de structure syntaxique, le deuxième l'attribution de la partie du discours). Nous utiliserons les termes analyseur et parseur interchangeablement.

2.1.3 L'apprentissage automatique

Pour la réalisation de nos analyses syntaxiques, nous utilisons des techniques qui appartiennent au domaine de l'apprentissage automatique.

L'apprentissage automatique : Ensemble de techniques et méthodes statistiques qui se basent sur des données observées pour entraîner automatiquement des modèles prédictifs. Certaines parties d'un modèle d'apprentissage automatique ne nécessitent pas de paramétrisation explicite de la part de son concepteur. Le terme a été employé la première fois par Arthur Samuel dans un article d'IBM en 1959 (Samuel 1959) pour désigner le processus d'apprentissage d'un algorithme capable de jouer au jeu de dames. Dans ce mémoire, nous utiliserons à la fois des techniques d'apprentissage supervisé et d'apprentissage non supervisé.

Apprentissage supervisé : Famille de techniques d'apprentissage automatique qui cherche à prédire en fonction de l'état d'un système (input) une représentation numérique discrète ou continue (output). Pour pouvoir s'entraîner, on donne au modèle des couples input/output préalablement annotés.

Apprentissage non supervisé : Famille de techniques d'apprentissage automatique qui traitent des données non annotées. Les modèles non supervisés ont souvent comme "objectif" de déterminer la structure des données ou de réduire la dimension de l'espace vectoriel des données.

Réseau de neurones artificiels : Un réseau de neurones artificiels (RNA) est un cas particulier de modèle d'apprentissage automatique. En 1943, le neurophysicien Warren McCullough et le mathématicien Walter Pitts ont donné une hypothèse sur le fonctionnement des neurones biologiques (McCulloch and Pitts 1943). Pour illustrer leurs propos, ils ont modélisé sur circuit imprimé un réseau de neurone artificiel. En 1958, Frank Rosenblatt inventa le perceptron, un réseau de neurones capable d'effectué des classifications binaires. Différents types de RNA ont par la suite vu le jour pour répondre à différents besoins comme les réseaux convolutifs (pour traiter des images), les réseaux récurrents (pour traiter les séquences temporelles), etc.
Le type le plus commun, le **RNA à propagation avant** (*feed-forward neural network*, FFNN), est composé de plusieurs couches successives de neurones reliées entre elles. Dans chaque couche, les neurones reçoivent en entrée des valeurs numériques de la part des neurones de la couche précédente afin d'y appliquer une combinaison linéaire ainsi qu'une fonction d'activation non linéaire. La valeur obtenue en sortie de chaque neurone est ensuite transmise aux neurones de la couche suivante. Les coefficients de chaque neurone sont optimisés lors de l'entraînement grâce à la technique de rétro-propagation du gradient d'erreur. Grâce à tout ce processus, les RNA sont capables de modéliser des problèmes non-linaires complexes.

Rétro-propagation : Algorithme d'optimisation qui permet d'entraîner les RNAs (Rumelhart et al. 1986). La rétro-propagation calcule le gradient d'erreur d'une prédiction avec la valeur attendue et propage ce gradient en remontant dans les couches pour calculer une approximation de la contribution de chaque neurone (poids) du modèle sur le gradient d'erreur (d'où le terme rétro-propagation). Ces poids sont modifiés proportionnellement à cette contribution calculée. On procède ensuite de manière itérative en appliquant ce procédé de rétro-propagation du gradient sur d'autres échantillons afin de réduire progressivement l'erreur du modèle[1].

Réseau de neurones récurrents (RNR) : Ils sont des cas particuliers de RNA qui peuvent prendre en compte la temporalité et garder une information pour les états observés et prédictions passées du système. Ceci est très utilisé pour l'analyse de séries temporelles ou de séquences d'unités linguistique.

2.2 Le naija

2.2.1 Un pidgin-créole à base anglaise

Le naija est un pidgin-créole de l'anglais qui descend du créole du sud du Nigeria dans les régions du delta du Niger. Ce pidgin, apparu lors de l'époque coloniale (fin du 19e siècle), est issu d'un mélange entre l'anglais, le portugais et des langues locales (igbo, yoruba, edo, etc...). Le naija a subi une rapide croissance de son adoption depuis l'indépendance du pays vis-à-vis du Royaume-Uni en 1960 (Caron 2009).

Bien que n'étant pas la langue officielle du pays (qui est l'anglais), le naija est utilisé par plus de 100 millions de Nigérians en tant que *lingua franca* dans des contextes généralement informels ou privés par les jeunes scolarisés ainsi que par l'élite. Il est la langue maternelle de quelques millions d'habitants (pas d'estimation connue).

À ce jour, peu de travaux sur la langue ont été réalisés bien que le naija se soit imposé auprès de la population. Une raison à cela est la mauvaise image qu'a le naija aux yeux des Nigérians qui le considèrent plus comme une version dégradée et populaire de l'anglais ("*broken English*") que comme une langue à part entière.

Le terme de *naija* pour désigner cette langue est controversé puisqu'il signifie l'adjectif *du nigeria* en pidgin du Nigeria et semble exclure le pidgin du Cameroun qui est une langue très proche du naija.

[1] Les vidéos sur ce sujet de *3blue1brown* sont très visuelles et informatives et nous ne pouvons que vous recommander la visualisation (lien vers la vidéo sur l'optimisation du gradient : `https://www.youtube.com/watch?v=IHZwWFHWa-w`)

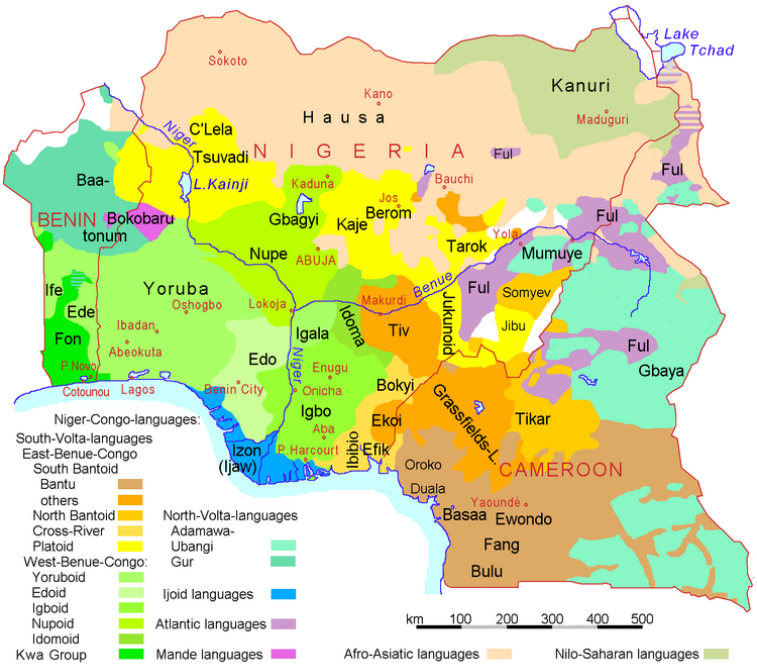

Figure 2.5: Répartition des langues et familles de langues présentes sur les territoires du Nigeria, du Bénin et du Cameroon (source : Ulamm, Wikipedia)

2.2.2 Langues substrats

Trois des substrats dominants du naija, à savoir le yoruba, l'edo et l'igbo sont géographiquement situés au sud du Nigeria et font partie des langues nigéro-congolaises. Cette famille de langues est celle la plus étendue d'Afrique en termes de locuteurs (450 millions).

Le quatrième substrat important, le hausa, est une langue afro-asiatique. Cette famille s'étend géographiquement sur l'Afrique du Nord, le Sahara, le Moyen-Orient ainsi que la corne de l'Afrique. Parlée par 410 millions de personnes et constituée de 310 langues, cette famille comprend notamment les sous-familles des langues berbères et sémitiques.

La figure 2.5 montre les différentes langues et familles de langues présentes sur les territoires du Nigeria, du Bénin et du Cameroon.

2.2.3 Attributs linguistiques

Similarités avec l'anglais :
L'anglais est la principale langue lexifiante du naija et donc de nombreux mots du naija sont empruntés de l'anglais. Il existe aussi des mots qui proviennent d'autres langues européennes (espagnol, portugais, français), ainsi que des langues locales.

Différences avec l'anglais :

Contrairement à l'anglais qui est une langue plutôt analytique, le naija est, comme beaucoup de pidgins, une langue isolante (c.-à-d. une langue qui possède peu d'affixes grammaticaux). (Courtin et al. 2018) observe que le naija contient des constructions de verbes sériels (SVC), aussi présents en yoruba. Un SVC est une "construction mono-clausale constituée de multiples verbes indépendants avec aucune conjonction de coordination et aucune relation prédicat-argument entre ces verbes" (Haspelmath 2016).

2.3 Les Treebanks

2.3.1 Définition

Les treebanks (Marcus et al. 1993; Hajic et al. 2001; Abeillé et al. 2003; Nivre et al. 2016) sont des corpus arborés qui comportent une information syntaxique ou sémantique sur la phrase (chaque phrase du corpus est accompagnée d'une annotation de sa structure syntaxique ou sémantique). Ils sont majoritairement utilisés dans deux domaines : en linguistique de corpus et en linguistique computationnelle.

En linguistique de corpus, les treebanks permettent de repérer de nouvelles constructions, de quantifier les constructions connues, d'étudier l'émergence d'une construction, etc. On peut à partir des treebanks extraire des lexiques syntaxiques et des grammaires formelles (O'Donovan et al. 2005).

En linguistique computationnelle, le treebank sert de ressource pour entraîner et évaluer des modélisations statistiques du langage avec par exemple des analyseurs syntaxiques (Charniak 1997; Yamada and Matsumoto 2003). Du fait de l'utilité de ces treebanks pour la linguistique computationnelle et de corpus, de nombreux projets voient le jour et s'ajoutent à la grande quantité de corpus annotés déjà existants. Nous allons présenter le plus important projet de co-développement de treebank, celui de Universal Dependencies.

2.3.2 Universal Dependencies (UD)

UD est un projet international et open-source débuté en 2014 par (Nivre et al. 2016) qui se base sur les travaux de Stanford Dependencies, Google universal POS tags et Interset Interlingua dans le but de créer un système universel et consistant d'annotations de treebanks en syntaxe de dépendance. Selon les auteurs, un tel système universel permettrait de faciliter les études typologiques entre les langues ainsi que la création d'outils de traitement du langage multilingues (comme des analyseurs syntaxiques, outils de traduction automatique, recherche d'entités nommées, etc. . .).

Puisque l'un des objectifs est d'entraîner des algorithmes d'apprentissage automatique, le formalisme choisi pour encoder l'information est la syntaxe de dépendance. Pour ce qui est du choix du gouverneur dans un couple de mots, UD se distingue de la majorité des grammaires

(de dépendances ou autre) existantes comme les grammaires génératives (syntaxe X-barre), la "Meaning-Text Theory" (Mel'čuk et al. 1988), la "Word Grammar" (Hudson 1984), etc... En effet, plutôt que de se baser sur un critère distributionnel pour étudier les dépendances, les auteurs de UD se fient aux deux règles suivantes :

- Les mots lexicaux (ceux qui apportent le contenu) sont reliés par des relations syntaxiques.
- Les gouverneurs des mots outils sont les mots lexicaux qu'ils introduisent.

En suivant ces règles, les constructions fonctionnelles spécifiques à une langue n'influent pas sur l'ossature principale de l'arbre contenant les mots outils. On a donc, en théorie, une homogénéisation basée sur la sémantique des structures arborées qui se produit entre les langues, et il devient alors possible de comparer ces langues ainsi que de créer des outils d'apprentissage automatique.

Au moment de publier ce mémoire, il existe 150 treebanks au format UD pour un total de 90 langues représentées qui ont tous été réalisés (ou convertis d'autres formats) lors de ces cinq dernières années. Ces treebanks sont utilisés pour de l'apprentissage d'outils monolingues (Oh et al. 2020), l'apprentissage d'analyseurs syntaxiques multilingues (Duong et al. 2015; Ammar et al. 2016; Muller et al. 2020) ainsi que des études typologiques (Chen and Gerdes 2020).

Il semblerait que les ressources de UD remplissent certaines des motivations du projet. Cependant, le choix majeur d'établir les dépendances autour des mots lexicaux ne fait pas consensus chez les grammairiens et l'on peut voir des travaux qui proposent des modifications pour améliorer le format des treebanks. (Osborne and Gerdes 2019) comparent le choix des "mots lexicaux" avec celui des "mots fonctions" quant aux six desiderata de UD. Selon eux, une approche plus classique basée sur la distribution permettrait de répondre mieux aux motivations de UD.

2.3.3 Syntactic-surface Universal Dependencies (SUD)

Dans l'objectif d'être plus fidèle à la syntaxe distributionnelle, (Gerdes, Guillaume, et al. 2018) proposent le projet SUD qui se veut être un nouveau système d'annotation complémentaire à UD. Alors que l'universalité de UD s'obtient en reliant d'abord entre eux les mots lexicaux, l'universalité de SUD s'obtient en se fiant à la distribution des syntaxèmes d'un syntagme pour déterminer les dépendances.

D'après les auteurs de SUD, ce nouveau paradigme d'annotation apporterait les avantages suivants :

- Il n'est plus nécessaire de déterminer la "lexicalité" d'un mot par rapport à un autre pour déterminer le gouverneur. Cette notion est en effet peu objective et deux annotateurs différents peuvent, dans un contexte similaire, annoter différemment en fonction de leur

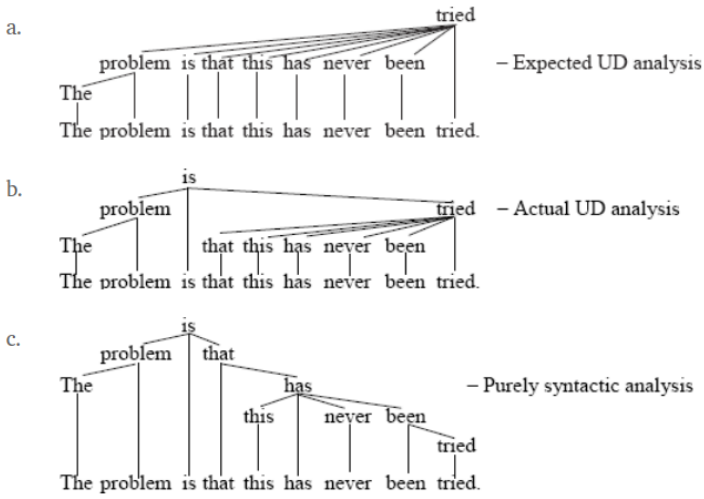

Figure 2.6: Différences entre l'annotation UD standard, l'annotation ponctuelle et l'annotation de SUD (Osborne and Gerdes 2019)

jugement sur le contenu que porte un mot (voir des exemples dans Gerdes and Kahane 2016)

- Dans certains cas, les règles d'annotations ne sont pas consistantes et possèdent des exceptions. Par exemple, dans le cas d'une clause prédicative en anglais (voir 2.6), l'annotation de UD n'est pas consistante avec la règle d'attachement du gouverneur basée sur la lexicalité d'un mot.

- La distance moyenne gouverneur-dépendant est plus petite. Ceci est en accord avec certaines théories psycholinguistiques sur la performance linguistique de (H. Liu 2008; H. Liu et al. 2017; Futrell et al. 2017), qui supposent que le langage tend à minimiser la longueur des liens syntaxiques pour réduire l'effort cognitif de traitement de l'énoncé.

Le projet se veut être une alternative à UD et non pas un concurrent. Les différents schémas d'annotations sont en théorie isomorphiques (la même information est présente mais encodée différemment) et les auteurs proposent d'utiliser les outils existants de réécriture d'arbres (Guillaume et al. 2012) pour convertir les structures d'un formalisme à l'autre. Le site Grew-Match [2], dédié à la requête de structure syntaxique de treebank, répertorie les treebanks natifs UD convertis au format SUD ainsi que les treebanks natifs SUD convertis au format UD. Il est donc possible d'obtenir une version SUD de chaque treebank au format UD et inversement. Le nombre de treebanks natifs SUD est pour le moment de quatre (deux treebanks du français oral, un treebank du français écrit et un treebank du naija oral (en cours de réalisation).

[2] `http://match.grew.fr/`

2.3.4 Le NaijaSynCor Treebank

Le projet ANR NaijaSynCor a pour but de créer le premier treebank annoté en prosodie et en syntaxe du naija oral. Ce projet de 42 mois[3] dirigé par Bernard Caron du Llacan (CNRS) récupère l'expertise ainsi que les outils développés sur le projet ANR Rhapsodie (Lacheret et al. 2014) qui a mené à la création du plus grand corpus annoté en prosodie et en syntaxe du français oral. Le treebank peut être utile pour développer des outils de TAL (comme présenté dans ce mémoire) mais aussi pour promouvoir l'identité de cette nouvelle langue en révélant des constructions uniques distinctes de l'anglais (Courtin et al. 2018).

2.3.5 Processus de création d'un treebank

Les projets de création de treebanks sont des processus longs qui s'étalent généralement sur plusieurs années et qui comportent plusieurs phases :

- Choix d'un schéma d'annotation (Gerdes and Kahane 2016)
- Choix du corpus. Si le corpus est oral, il faudra utiliser une méthode (automatique, manuelle, mixte) pour transcrire le texte.
- Choix des jeux d'étiquettes correspondant aux différents traits et propriétés pertinentes pour la modélisation (gouverneur, relation syntaxique, lemme, partie du discours, glose, flexion, etc).
- Création du guide d'annotation. Il est très important que les annotateurs restent cohérents entre eux tout au long du projet si l'on veut pouvoir espérer d'en tirer des mesures intéressantes ou encore que l'analyseur syntaxique généralise correctement.
- Pré-annotation automatique de l'analyseur pour aider l'annotateur dans sa tâche. Au début des projets, l'analyseur automatique est peu performant et n'apporte donc que peu d'aide. Cependant, avec les itérations successives ("bootstrapping"), la performance de l'analyseur s'améliore et fournit toujours plus d'aide aux annotateurs.
- Correction manuelle par les annotateurs. Cette annotation se fait généralement par lots de phrases. Après annotation d'un lot, l'analyseur est ré-entraîné afin d'affiner la modélisation. L'accord inter-annotateur est calculé sur ces corrections pour évaluer la cohérence des annotations entre les annotateurs. A ceci peut aussi s'ajouter des méthodes de fouille d'erreurs grâce a des outils statistiques (nous en utilisons certains dans la partie 5).
- Publication du treebank, mise en disponibilité d'outils de requêtes, analyses syntaxiques basées sur le treebank.

[3] février 2017 -> septembre 2020, prolongé jusqu'à l'été 2021

2.4 L'état de l'art

2.4.1 Le traitement automatique des langues (TAL)

Le traitement automatique des langues (TAL), que nous considérons ici comme l'équivalent du terme anglais "Natural Language Processing" (NLP), a vu le jour au milieu du 20e siècle dans les années 40 en partie avec les réflexions de (Weaver 1949) dans son mémorandum ainsi que les investigations de Booth et Richens. Il était question à l'époque d'essayer de trouver une solution au problème de traduction automatique qui semblait être réalisable grâce aux récentes avancées en informatique. On y trouve déjà des questionnements sur l'utilité d'étudier les mots dans leurs contextes afin de pouvoir les désambiguïser. Ce concept, popularisé quelques années plus tard par le linguiste John Rupert Firth en 1957 avec sa célèbre citation "You shall know a word by the company it keeps" (Firth 1957), est un des principes fondamentaux des techniques de plongements vectorielles que nous utilisons aujourd'hui.

Cette première période, empiriste, a été suivie par une période plus rationaliste de 1960 à 1985 inspirée par la grammaire générative de Chomsky. Cette linguistique se pose des questions sur la grammaticalité et est plus intéressée par la question de si un énoncé est grammatical ou non plutôt que de se demander s'il est fréquent ou non dans les corpus.

Dans les années 1990, l'arrivée massive de données lisibles par la machine ainsi que l'amélioration continue de la puissance de calcul des ordinateurs ont permis un développement important des méthodes statistiques pour traiter le langage. On y voit là un retour des méthodes et idées empiristes des années 1950. La création et l'évaluation de ces méthodes statistiques ne peut se faire que grâce à l'utilisation de données contrôlées ce qui amène un élan de création de projet de corpus en tout genre (corpus parallèles, treebank, corpus de désambiguïsation, etc...). C'est notamment dans cette décennie que les premiers treebanks voient le jour avec parmi eux l'English Penn Treebank (Marcus et al. 1993) qui fut le premier treebank à grande échelle. Grâce à ces nouveaux treebanks annotés syntaxiquement, de nouvelles méthodes pour analyser la structure syntaxique d'un énoncé ont émergé.

2.4.2 L'analyse syntaxique automatique (parsing syntaxique)

L'analyse syntaxique automatique consiste à attribuer automatiquement la structure syntaxique d'un énoncé, au départ à l'aide de grammaires formelles écrites à la main, et aujourd'hui grâce à l'utilisation de modèles statistiques ou symboliques.
Cette tâche est généralement utilisée en amont d'autres tâches de TAL telles que l'extraction d'information (Miyao et al. 2008) ou les systèmes de questions-réponses (Reddy et al. 2014). Les analyseurs syntaxiques statistiques utilisent généralement comme ressources d'entraînement et d'évaluation les treebanks développés par les linguistes. Le type de grammaires extraites par un analyseur dépendra donc du schéma d'annotation utilisé pour le treebank. Puisque le schéma utilisé par le projet NaijaSynCor est basé sur la dépendance, nous restreignons notre état de

l'art aux techniques utilisées pour les analyseurs en dépendances syntaxiques.

Pour bien comprendre le fonctionnement du parsing syntaxique en dépendance, il faut bien dissocier les deux opérations que le parseur doit effectuer :

- Construire l'arbre des dépendances syntaxiques
- Prédire les fonctions syntaxiques associées

La deuxième opération est la moins complexe en termes de modélisation puisque cela revient généralement à faire de la prédiction de classes dans un inventaire prédéfini. La première opération, quant à elle, nécessite de reconstruire une structure syntaxique en trouvant pour chaque syntaxème de la phrase son gouverneur ainsi qu'en respectant certains critères au niveau de la structure (arborescence du graphe, projectivité ou non, etc. . .).

Dans le but de résoudre ces objectifs, deux approches ont cohabité ces vingts dernières années : l'approche transitionnelle et l'approche par graphe.

2.4.2.1 L'approche transitionnelle (transition-based approach)

L'approche par transition introduite en 2003 par [Yamada et al., 2003] est reprise et améliorée à plusieurs reprises (Zhang and Nivre 2011; Dyer et al. 2015, etc.). Cette approche est la moins complexe algorithmiquement des deux alternatives ce qui a favorisé son développement et son utilisation. Cependant, depuis l'arrivée récente des techniques d'apprentissage profond (voir notre explication sur les transformers dans la sous partie suivante), l'approche par graphe est avantagée.

2.4.2.2 L'approche par graphe (graph-based approach)

Les travaux de McDonald (McDonald, Crammer, et al. 2005) ont introduit une méthode basée sur les graphes pour effectuer le parsing syntaxique.

L'approche par graphe peut être décomposée en deux étapes :

1. Un modèle qui assigne un score de connexion syntaxique à chaque couple de mots de la phrase.
2. Un algorithme qui, à partir de cette matrice de score, cherche à reconstruire l'arbre dirigé qui maximise le score de la structure.

1) Création de la matrice de scores

Le but du modèle est d'assigner des scores plus ou moins élevés aux différents couples de mots

de l'énoncé. Avec ces probabilités, on construit une matrice de taille $N \times N$ (avec N nombre de tokens (mot, ponctuation et ancre de la phrase) avec en i-eme ligne et j-eme colonne le score de gouvernance de w_i sur w_j. En théorie, il est possible d'avoir un modèle qui prédit la probabilité des N^2 arcs en même temps et donc le score assigné à un couple w_i/w_j influe sur tous les autres scores en même temps. Mais en pratique, l'algorithme de résolution d'un tel système a une complexité de calcul trop élevée et n'est pas solvable en un temps raisonnable (le problème est dit NP difficile). On a donc seulement un calcul de premier ordre des scores qui calcule localement les probabilités. On suppose donc, à tort, que les arcs sont indépendants les uns des autres pour se réduire à un problème de premier ordre (c.-à-d. le score de chaque couple est calculé localement et ne prend pas en compte les autres scores lors de son calcul).

Pour calculer ces scores locaux, plusieurs algorithmes d'apprentissage automatique se sont succédés. (McDonald, Crammer, et al. 2005) utilisent MIRA (*margin infused relaxed algorithm* (Crammer and Singer 2001)) pour calculer cette matrice. En 2016, les travaux de (Kiperwasser and Goldberg 2016) présentent un parseur qui utilise le mécanisme d'attention nouvellement introduit par (Bahdanau et al. 2014) pour de la traduction neuronale. Ces travaux ont grandement inspiré (Dozat and Manning 2016) pour la création de leur architecture d'attention bi-affine. C'est une variation de cette structure détaillée dans la section 3.2 que nous allons utiliser.

2) Construction de l'arbre syntaxique à partir de la matrice de probabilités des gouverneurs

Cette partie de l'approche par graphe est plus "triviale" et est résolue depuis un des premiers papiers de McDonald qui traite la construction d'arbres projectifs ET non projectifs depuis une matrice de scores (McDonald, Pereira, et al. 2005). En fonction de si l'on veut permettre à notre modèle de reconstruire des arbres projectifs ou non, il faudra choisir entre les deux algorithmes suivants : l'algorithme d'Eisner pour les structures projectives (Eisner 1997) l'algorithme de Chu-Liu-Edmond (CLE) pour les structures non projectives (Chu 1965; Edmonds 1967)

L'algorithme d'Eisner possède l'avantage de pouvoir reconstruire des arbres projectifs pour les langues le nécessitant, mais cela vient avec un coût algorithmique plus élevé ($O(n^3)$) que l'algorithme de Chu-Liu-Edmond ($O(n^2)$). Pour nos travaux, nous avons utilisé l'algorithme de CLE puisque le naija possède quelques (rares) constructions non projectives. Nous détaillons un peu plus en détail l'algorithme en section 3.2.

2.4.3 Les plongements lexicaux (words embedding)

Les plongements lexicaux sont basés sur l'hypothèse distributionnelle (Harris 1951; Firth 1957) qui suppose que des mots qui apparaissent dans des contextes similaires ont des sens proches. C'est dans cette optique que les plongements lexicaux cherchent à associer à chaque mot un vecteur qui encode le contexte d'apparition de ce mot. Encoder l'information avec des vecteurs permet de pouvoir calculer des similarités ou des distances (syntaxique, sémantique, etc.) entre les vecteurs. Il est possible d'attribuer, par le biais d'algorithme, des vecteurs proches pour

des unités ayant des signifiés similaires et inversement des vecteurs éloignés pour des unités aux signifiés différents.

2.4.3.1 Plongement statique

Les plongements lexicaux ont longtemps été "statiques", c'est-à-dire que la technique d'obtention des vecteurs ne permet pas de prendre en compte la polysémie d'un mot. Par exemple, si les lexies avocat[fruit] et avocat[profession] sont présents dans un même texte et qu'on applique une méthode de plongement lexical statique, alors les deux homophones sont représentés par un même vecteur qui encode de l'information des deux contextes à la fois (dans une proportion qui dépend en partie du rapport d'occurrence des deux homophones). Cette difficulté à discerner le contexte ne concerne pas seulement les polysèmes mais aussi les mots ayant plusieurs fonctions syntaxiques en fonction du contexte.

Bien que les plongements lexicaux soient utilisés depuis les années 90 avec notamment les techniques d'analyse sémantique latente (Landauer and Dumais 1997), l'avancée la plus remarquable des plongements statiques est sans doute l'invention de word2vec par (Mikolov et al. 2013) qui s'appuie sur les réseaux de neurones artificiels (RNA) pour modéliser le contexte. L'idée générale est d'entraîner un RNA (de manière non supervisée !) à prédire un mot cible en fonction des mots présents dans une fenêtre de taille paramétrable autour du mot cible (voir 2.7). Une fois l'entraînement fini (sur des corpus généralement très volumineux), la représentation cachée du RNA obtenue lors de la prédiction d'un mot est récupérée pour devenir le plongement vectoriel de celui-ci.

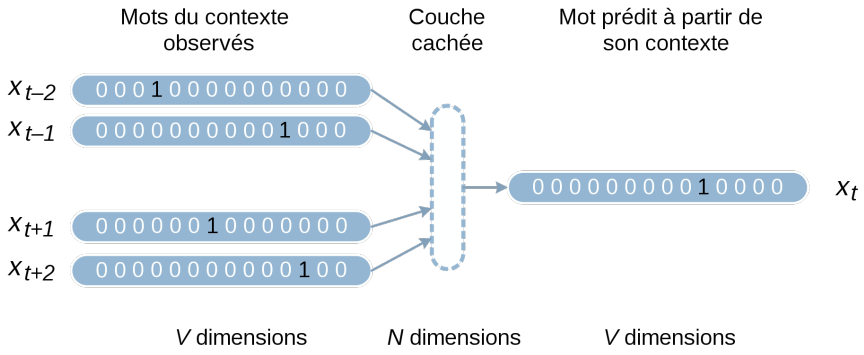

Figure 2.7: Exemple d'utilisation d'un RNA pour la prédiction d'un mot (w_t) en fonction de son contexte (fenêtre de 5 mots). La représentation vectorielle du mot w_t est la représentation cachée des neurones obtenus après optimisation lors de l'entraînement. (figure de Jarez Renz, wikipedia)

2.4.3.2 Plongement contextuel

Grâce aux progrès récents de l'informatique, aussi bien du côté du matériel (cartes graphiques, processeurs, etc.) que du côté des techniques de deep learning, nous avons pu voir émerger ces

cinq dernières années des modèles de plongements vectoriels contextuels. Pour effectuer cette contextualisation, des couches de réseaux récurrents (RNN) telles que des LSTM (Long-Short Term Memory) sont ajoutées en sortie de l'embedding statique dans le but d'ajouter à chaque mot une information sur les autres mots présents dans le contexte.

Si l'on reprend notre exemple avec le polysème "avocat", le vecteur pour avocat[je mange mon avocat avec du sel] sera différent de avocat[je suis allé voir mon avocat]. Dans une chaîne d'un LSTM (Long-Short Term Memory), l'information se propage dans le sens de la phrase ou dans les deux sens en fonction de si on utilise un réseau récurrent unidirectionnel (p. ex. LSTM) ou bidirectionnel (p. ex. bi-LSTM). Dans le cas d'une propagation unidirectionnelle, un token ne possède pas l'information des tokens lui succédant alors que pour une propagation bidirectionnelle, chaque token de la phrase absorbe de l'information de tous les autres mots de la phrase.

2.4.4 L'apprentissage par transfert

L'apprentissage par transfert est une technique d'apprentissage automatique qui consiste à utiliser les connaissances apprises sur un problème source pour s'aider dans la résolution d'un problème cible (voir 2.8).

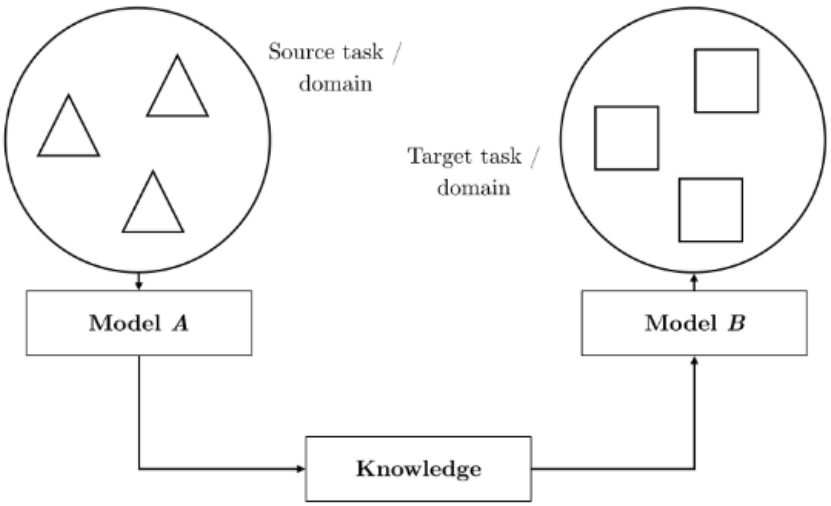

Figure 2.8: Schématisation du principe d'apprentissage par transfert. Les connaissances apprises sur A sont utilisées sur la problématique B. (Ruder 2019)

L'apprentissage par transfert est particulièrement utile lors de l'entraînement de modèle pour des tâches qui sont peu dotées en données. Le moyen de transfert le plus courant à l'heure actuelle est via le partage partiel ou complet de la représentation interne d'un réseau de neurones.

Il existe pour le moment deux grandes familles d'apprentissage par transfert (AT) : l'AT transductif et l'AT inductif :

- L'apprentissage par transfert transductif est utilisé lorsque l'on crée un modèle sur des données d'un domaine source A puis que l'on transfère sur des données d'un domaine cible B où l'on ne possède pas les annotations des observations. L'enjeu est alors d'utiliser les observations du domaine B à bon escient afin de construire, de manière non supervisée, une modélisation de la distribution des échantillons (Arnold et al., 2007). Pour cette famille d'apprentissage, les tâches cibles et sources doivent être identiques et seule la distribution des données peut changer (p. ex. changement du domaine d'application d'un modèle de recherche d'entités nommées).

- L'apprentissage par transfert inductif, quant à lui, est utilisé lorsque l'on connaît les annotations associées aux observations du second domaine. Le but est alors d'utiliser au mieux les connaissances apprises sur le domaine A pour affiner la modélisation de B. Pour cette famille, non seulement le domaine d'application peut changer (anglais -> français ; textes journalistiques -> textes oraux) mais aussi la nature de la tâche (prédiction du mot suivant -> prédiction de la partie du discours des mots de la phrase).

Nos travaux se situent dans ce second cas d'apprentissage par transfert inductif puisque nous disposons de couples d'observations/annotations pour notre domaine cible (treebank du naija).

Les techniques appartenant à l'apprentissage par transfert inductif peuvent encore être divisées en deux sous-familles (voir 2.9, (Ruder 2019)), celles ou les tâches sont entraînées simultanément, et celles où les tâches sont entraînées séquentiellement. L'approche simultanée est favorisée lorsque le volume de données est plus ou moins égale entre les différentes tâches afin de maximiser le transfert entre les tâches. Inversement, lorsque la tâche source possède un nombre de données supérieur de plusieurs ordres de grandeur par rapport à la tâche cible, l'approche séquentielle est utilisée afin d'éviter que le modèle ne se focalise trop sur la tâche majoritaire.

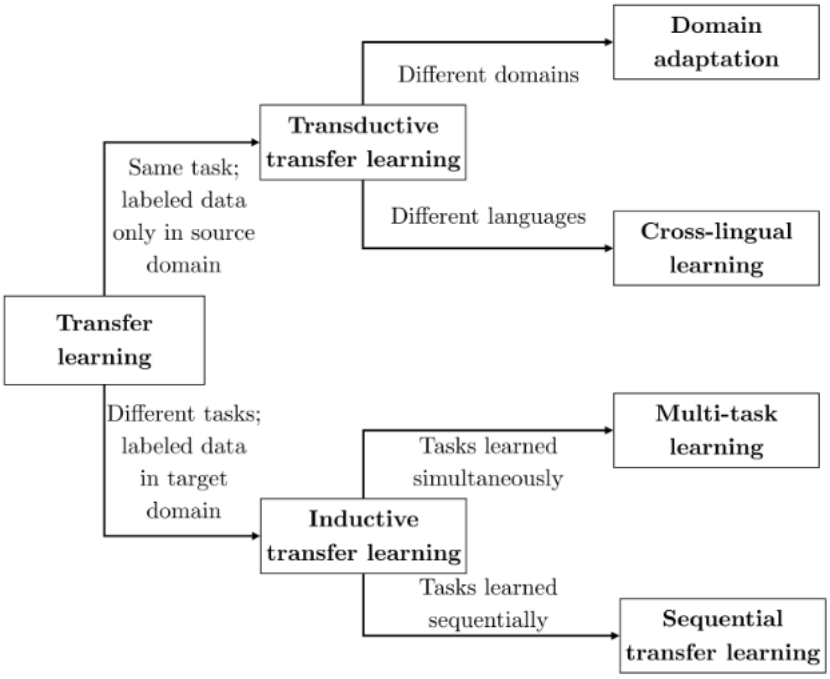

Figure 2.9: Taxonomie de l'apprentissage par transfert. (Ruder 2019)

L'apprentissage par transfert séquentiel

Lors de l'apprentissage par transfert séquentiel, un modèle est d'abord entraîné sur une tâche source avant d'être utilisé pour l'entraînement d'une tâche cible. La première phase est généralement appelée la phase de pré-entraînement alors que la seconde est appelée la phase d'adaptation (Ruder 2019). Lors de la phase d'adaptation, deux stratégies existent : le feature-extraction et le fine-tuning.

Le **feature-extraction** consiste à geler les poids du modèle pré-entraîné et de s'en servir comme entrée d'un nouveau modèle qui, lui, sera optimisé. Il y a donc une extraction de la représentation interne du modèle gelé qui sert d'entrée à un autre modèle.

Le **fine-tuning**, inversement, permet la modification des coefficients du modèle pré-entrainé lors de l'optimisation. Le modèle pré-entraîné sert donc d'initialisation pour le modèle de la tâche cible.

Le choix d'une approche par rapport à l'autre se fait en fonction de la similarité entre les tâches. (Peters et al. 2019) montrent dans leurs travaux que les deux approches se valent généralement, le fine-tuning prenant le dessus lorsque les tâches sources et cibles sont similaires alors que le feature-extraction à l'avantage lorsque les tâches sont différentes. Cependant, leurs travaux montrent aussi que différents types de modèle de langue (ELMo et BERT) ont des comportements différents.

Les techniques de plongement lexicaux peuvent génralement être utilisés pour l'apprentissage par transfert puisqu'elles permettent d'associer les mots de la langue à des représentations vectorielles apprises sur de grand corpus. Les modèle de langue (section suivante) utilisent aussi l'apprentissage par transfert pour transférer des connaissances plus profondes de la langue entre différentes tâches.

2.4.5 Les modèle de langue

Les transformers

Les transformers ont été inventés par (Vaswani et al. 2017). Jusque-là, pour obtenir une représentation vectorielle d'une séquence, des RNNs étaient généralement utilisés [4]. La structure séquentielle des RNNs fait que cette représentation n'est pas efficace pour les longues séquences (l'information étant encodé dans un seul vecteur de taille fixe, elle se retrouve diluée un peu plus à chaque nouveau mot).
C'est ce que propose de corriger (Bahdanau et al. 2014) en ajoutant un **mécanisme d'attention**[5] jointement avec un RNN. Ces mécanismes d'attention permettent au réseau d'apprendre pour chaque mot à focaliser son attention sur les autres mots pertinents (pour une tâche donnée).
(Vaswani et al. 2017) décident d'aller encore plus loin et de s'affranchir complètement des RNNs pour ne garder seulement que des mécanismes d'attention pour traiter la phrase. Dans le but d'effectuer de la traduction automatique, ils inventent le transformer, un réseau qui encode vectoriellement chaque mot d'une séquence en fonction des autres mots présents pour ensuite décoder cette séquence de vecteurs à l'aide d'un autre mécanisme d'attention.
Contrairement aux RNNs qui propagent l'information séquentiellement, les transformers la propagent en parallèle ce qui permet de paralléliser au mieux les calculs au niveau des cartes graphiques et donc de réduire le temps d'entraînement d'un modèle. Les transformers qui modélisent la langue sont donc entraînés plus longtemps et sur plus de données que les précédents RNNs et atteignent des résultats sans précédent sur la plupart des tâches de TAL (Brown et al. 2020).
Bien que les transformers soient pour le moment majoritairement utilisés dans le domaine du TAL, d'autres domaines commencent à adapter des architectures de transformers. On peut notamment citer le récent article de (Carion et al. 2020) qui propose d'utiliser un transformer pour faire de la détection d'objets.

Pour ce qui est du traitement automatique des langues, on peut catégoriser les transformers en deux parties : les encodeur-décodeurs et les encodeurs.
Les encodeur-décodeurs sont composés de couches encodeurs et décodeurs. Les encodeurs vont extraire l'information de la séquence sous forme de vecteurs tandis que les décodeurs vont partir

[4] Des réseaux convolutifs (CNN) peuvent aussi être utilisés mais dans une moindre mesure. Les CNNs sont surtout utilisés pour de l'analyse de données 2D, 3D, etc.

[5] Pour une explication plus approfondie du mécanisme d'attention, il peut être utile de se réferer au papier "Attention is all you need" (Vaswani et al. 2017) ainsi qu'à la vidée de Yannic Kilcher (`https://www.youtube.com/watch?v=iDulhoQ2pro`). La vidéo de Andrew Ng (`https://www.youtube.com/watch?v=quoGRI-1l0A`) peut aussi s'averer être une bonne explication de l'attention.

de cette représentation vectorielle pour produire une nouvelle séquence. Ils sont donc utilisés lorsque l'on veut que le produit final soit sous format textuel (modèle de génération de texte (GPT-3) pour du résumé de textes ; modèle de traduction, etc.).
Les encodeurs purs, quant à eux, sont composés uniquement de couches encodeurs. Ces modèles (BERT, RoBERTa, etc. . .) sont entraînés sur une ou plusieurs tâches leur permettant d'acquérir une connaissance statistique du langage. Ce qui rend ces modèles si attractifs est qu'il est très facile, une fois le pré-entraînement fini, de transférer les paramètres appris du modèle sur d'autres tâches de TAL (recherche d'entités nommées, calcul de similarité, parsing syntaxique, etc.).

Lors de nos travaux, nous avons utilisé des versions pré-entraînées d'encodeurs purs du type de BERT.

BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2018) est un type de transformer qui est composé seulement de couches encodeurs. Le nombre de couches peut varier en fonction de la complexité du modèle, généralement entre 12 pour BERT et 24 pour BERT-Large. Chaque couche est composée d'un mécanisme d'attention et d'un réseau de neurones à propagation avant (i.e. *feed forward neural network ; FFNN).* Le mécanisme d'attention prend en entrée chaque vecteur de la séquence encodé par l'encodeur précédent (ou par des plongements vectoriels statiques des mots dans le cas du premier encodeur de l'architecture) et transforme ces vecteurs en y ajoutant des informations "pertinentes" des autres vecteurs. Ce mécanisme de sélection de l'information pertinente est effectué à l'aide de modules d'attention qui, grâce à des opérations vectorielles entre différentes projections des vecteurs de la séquence, filtre l'information. La théorie mathématique derrière ce mécanisme d'attention est expliquée plus en détail dans l'article de (Vaswani et al. 2017). Un réseau perceptron récupère alors les vecteurs en sortie du mécanisme d'attention pour y appliquer des fonctions linéaires avant de transmettre l'information à l'encodeur suivant. Comme pour tous réseaux de neurones, BERT optimise ses paramètres lors de la phase d'entraînement à chaque rétro-propagation du gradient de l'erreur.

Le pré-entraînement de BERT se fait de manière non supervisée en ajoutant en sortie du dernier encodeur un classifieur softmax [6] dont la tâche est de prédire des mots masqués dans une phrase (ce qu'on appelle Masked Language Modeling (MLM)). C'est lors de cet entraînement que les coefficients des différents mécanismes d'attention et perceptrons vont s'optimiser à chaque itération de la rétropropagation du gradient d'erreur. Puisque cette tâche ne requiert pas d'annotation au préalable, il est possible d'entraîner BERT sur de très grands corpus. BERT est entraîné sur l'intégralité de Wikipédia ainsi que plusieurs milliers de livres ce qui représente environ 4 Go. Si cela semble impressionnant, ce n'est en fait qu'une petite partie de ce que d'autres modèles ont utilisé par la suite. RoBERTa, une des variantes de BERT, est entraîné sur 100 Go de données (1 Go est environ équivalent à 10 millions de mots), pour la plupart aspirées du web (Y. Liu

[6] La fonction softmax normalise les exponentielles des composantes d'un vecteur selon la formule suivante (pour un vecteur x avec N composantes) : $\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$

et al. 2019).

Une fois les modèles pré-entraînés, ils sont utilisés et adaptés sur d'autres tâches grâce à l'apprentissage par transfert (voir section 2.4.4). Il est possible ainsi d'adapter rapidement les représentations apprises sur de grands corpus bruts pour créer des modèles de TAL (NER, parsing, extraction d'informations, système de questions-réponses, etc...) sur des corpus spécifiques à la tâche (données annotées). Un des grands succès des modèles du type de BERT est justement cette capacité à pouvoir transférer efficacement la représentation acquise de ces grands corpus.

Ces modèles apprennent une certaine représentation de la langue grâce aux régularités statistiques présentes dans les corpus d'entraînement. Un nouveau champ d'étude, la bertologie, vise à comprendre la raison de l'efficacité de ces nouveaux modèles. Cependant, ce n'est pas toujours aisé d'autant plus que ces modèles disposent de toujours plus de paramètres. La version normale de BERT dispose de 110 millions, le modèle GPT-3, entrainé par facebook en début d'année 2020 en a 175 milliards (Brown et al. 2020) et les chercheurs de Google viennent de publier un article présentant une technique pouvant entraîner des modèles de 600 milliards de paramètres en quelques jours seulement sur 248 GPU (Lepikhin et al. 2020). Il est donc difficile de savoir avec exactitude quelles connaissances sont stockées dans les modèles, il semblerait qu'ils apprennent certains aspects de la structure du langage (Jawahar et al. 2019) comme les phénomènes de coréférences (Clark et al. 2019). Il semblerait même que ces différentes tâches de traitement (POS tagging, parsing, NER, semantic roles, coreference) soient apprises dans cet ordre au fil des couches d'encodeurs (Tenney et al. 2019).

Ces modèles sont généralement pré-entraînés par des entreprises en informatique (Google, Facebook, etc.) et des instituts de recherche (INRIA, KBLab/Sweden Library, etc.) qui disposent de capacités de calculs suffisamment grandes pour ensuite être mis à disposition librement sur différentes plateformes telles que Transformers d'HuggingFace[7].

[7] `https://github.com/huggingface/transformers`

Chapitre 3

Méthodologie

Ce chapitre a pour but de présenter le cadre pratique des expériences et analyses effectuées. Nous présentons dans la section 3.1 des informations sur le corpus (taille, format des données, schéma d'annotation suivi, séparation des données). La section 3.2 décrit l'architecture du modèle entraîné selon la procédure d'entraînement décrite dans la section 3.3.

3.1 Données

3.1.1 Le format Conllu

Nous avons travaillé sur un treebank annoté en syntaxe (et ultérieurement en prosodie) du naija oral : le NaijaSynCor Treebank (NST). Ce corpus est composé de 125k tokens annotés manuellement ainsi que de 350k tokens annotés automatiquement. Les 125k tokens ont été pré-annotés par bootstrapping grâce à l'intervention de notre parseur pour être ensuite corrigés par les annotateurs du projet NST.

Le NST est au format conllu qui est une convention créée par le projet Universal Dependencies pour s'assurer d'une homogénéité des différents treebanks annotés en syntaxe de dépendance. Les informations pour une phrase sont décomposées en deux parties.

Les métadonnées sont présentes en première partie à la suite des croisillons #. Les métadonnées disponibles dépendent des différents treebanks. Dans le cas du NST, nous pouvons retrouver dans ces métadonnées l'identifiant de la phrase, l'url de l'enregistrement audio, l'identifiant du locuteur qui l'a prononcée, le texte orthographique en naija, le texte comportant les balises macrosyntaxiques (en naija) et le texte traduit en anglais.

Les données internes à la phrase sont positionnés après les métadonnées sous format tabulaire de n lignes et 10 colonnes (n représentant le nombre de tokens de la phrase). Chacune de ces colonnes contient une propriété du token en question. Nous pouvons voir sur la figure 3.1 un

# sent_id = ABJ_GWA_03_Cost-Of-Living-In-Abuja_MG__14								
# sound_url = http://www.tal.univ-paris3.fr/trameur/iTrameur-naija/mp3/ABJ_GWA_03_Cost-Of-Living-In-Abuja_MG.mp3								
# speaker_id = Sp3								
# text = na here >+ I grow up //								
# text_en = This is where I grew up.								
# text_ortho = Na here, I grow up.								
1	na	na	PART	_	PartType=Cop	0	root	_	AlignBegin=24387|AlignEnd=24597|Gloss=be
2	here	here	ADV	_	_	1	comp:pred	_	AlignBegin=24597|AlignEnd=24705|Gloss=here
3	>+	>+	PUNCT	_	_	5	punct	_	AlignBegin=24705|AlignEnd=24735|Gloss=PUNCT
4	I	I	PRON	_	Case=Nom|Numbe	5	subj	_	AlignBegin=24735|AlignEnd=24827|Gloss=NOM.SG.1
5	grow	grow	VERB	_	_	1	comp:cleft	_	AlignBegin=24827|AlignEnd=25164|Gloss=grow
6	up	up	ADP	_	_	5	compound:prt	_	AlignBegin=25164|AlignEnd=25407|Gloss=up
7	//	//	PUNCT	_	_	1	punct	_	AlignBegin=25407|AlignEnd=25437|Gloss=PUNCT

Figure 3.1: Exemple d'une table conllu NST pour la phrase *Na here, I grow up*

exemple d'une phrase du treebank sous format conllu. Les informations importantes pour notre tâche sont :

- La position du token dans la phrase (colonne 1)
- La forme du token (colonne 2)
- La position du gouverneur du token (colonne 7)
- La relation syntaxique que le token possède avec son gouverneur (colonne 8)

Nous avons décidé de ne pas utiliser l'information sur le lemme (colonne 3) et la catégorie du discours (colonne 4) des tokens car ceux-ci ne sont pas accessibles lors de l'utilisation du parseur en conditions réelles sur du texte brut.

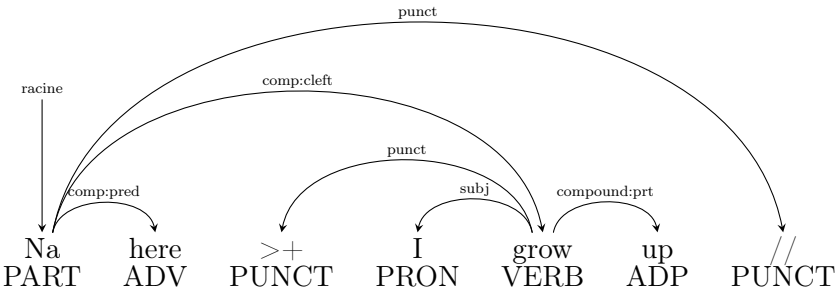

Figure 3.2: Arbre de dépendance syntaxique linéarisé et avec ajouts d'étiquettes morpho-syntaxiques et des marqueurs macro-syntaxiques pour l'énoncé *Na here, I grow up* (anglais : *This is where I grew up*)

Le schéma d'annotation suivi pour le projet NaijaSynCor est celui de SUD (voir section 2.3.3). Ce système d'annotation suit des règles précises quant à la décision des gouvernances et des relations syntaxiques de la phrase. Ces règles sont disponibles sur le site de SUD [1] et nous vous recommandons de vous y référer si besoin.

[1] `https://surfacesyntacticud.github.io/guidelines/u/`

3.1.2 Séparation du jeu de données

Pour éviter au maximum tout biais d'apprentissage lors de l'entraînement et la comparaison des modèles, nous avons effectué une séparation des données en trois jeux : jeu d'entraînement (80%), jeu de validation (10%) et jeu d'évaluation (10%). Les modèles sont entraînés sur le jeu d'entraînement et après chaque *époque*[2], on calcule la performance des modèles sur le jeu de validation pour observer si les modèles continuent d'apprendre correctement. Lorsque le modèle arrête d'apprendre pour un certain nombre d'époques (défini par l'utilisateur), les modèles sont évalués sur le jeu d'évaluation.

Pour effectuer la séparation, plutôt que d'effectuer une répartition aléatoire au niveau de la phrase, nous effectuons une répartition aléatoire au niveau de l'échantillon. Le treebank est divisé en 80 échantillons où chacun retranscrit le discours d'une ou plusieurs personnes sur un thème particulier. De ce fait, si l'on mélange aléatoirement au niveau des échantillons, la distribution des données de développement et de tests s'éloignent des données d'entraînement ce qui peut les rapprocher des données réelles.

Nous avons donc effectué différentes séparations en fonction des besoins. Les expériences des sections 4.1, 4.2 et 4.3 ont utilisé les données de la séparation 1. L'expérience de la section 4.4, plus particulière, a utilisé les données de la séparation 2. Enfin, l'analyse des prédictions de l'analyseur du chapitre V se base sur les prédictions de modèles entraînés grâce à de la validation croisée.

3.2 Architecture du modèle

Pour effectuer l'analyse des dépendances syntaxiques, nous utilisons l'architecture de (Dozat and Manning 2016) en remplaçant les couches de Bi-LSTM initiales par un modèle BERT. La sortie de la dernière couche du transformer BERT est reliée à l'entrée d'un MLP (Perceptron Multi Couches) dont le but est de réduire la dimension de chaque vecteur contextualisé de la séquence avant d'appliquer la transformation bi-affine. Nous pouvons retrouver cette architecture dans les travaux suivants (Oh et al. 2020; Kondratyuk and Straka 2019; Muller et al. 2020).

3.2.1 Segmentation en sous-mots

La première étape du traitement est la segmentation des tokens en sous-mots qui permet d'avoir un ensemble fini relativement restreins d'éléments de départ. Pour réaliser ce vocabulaire de sous-mots, on calcule au préalable les N sous-mots les plus fréquents du corpus qui constitueront par la suite les plus petites unités insécables du modèle. Le choix du segmenteur n'est pas contraint et l'on a par exemple une utilisation du segmenteur WordPiece (Schuster and Nakajima 2012)

[2]Une époque consiste à un passage entier sur un jeu de données pour l'apprentissage. Si l'on effectue 10 époques, cela revient à dire que le modèle a utilisé 10 fois le jeu de données pour son apprentissage.

pour les modèles BERT de Google et le segmenteur BPE (Sennrich et al. 2015) pour les modèles RoBERTa faits à Facebook. La taille du vocabulaire n'est aussi pas contrainte et son choix dépend de plusieurs facteurs comme par exemple la taille du corpus d'entraînement. Pour donner un ordre d'idée, la taille du vocabulaire du segmenteur associé au modèle BERT-anglais est de 42.000 sous-mots alors que la taille du vocabulaire du segmenteur associé au modèle BERT-multilingue est de 110.000 sous-mots.

3.2.2 Contextualisation vectorielle (BERT)

La séquence de sous-mots est ensuite donnée en entrée du transformer BERT (Bidirectional Encoder Representations from Transformers, (Devlin et al. 2018), voir notre déscription de BERT en 2.4.5) qui calcule une représentation vectorielle contextualisée de ces sous-mots dans la séquence. Il existe plusieurs variantes de BERT réalisées par différents groupes de chercheurs. Les propriétés et attributs des modèles (taille, corpus d'entraînement, etc...) dépendent de la variante. Voici un descriptif des deux modèles que nous avons le plus utilisés lors de nos recherches :

3.2.2.1 BERT anglais

Auteur(s)	Devlin et al.
Année	2018
Langue(s)	Anglais
Taille	Normal (110M de paramètres) / Large (340M de paramètres)
Tâche(s) d'entraînement	MLM + prédiction de la phrase suivante
Corpus d'entraînement	BookCorpus (800M de mots) et Wikipedia-Anglais (2500M de mots) 4 Go
Segmentation (sous-mots)	Wordpiece (vocabulaire de 42k sous-mots)

3.2.2.2 BERT multilingue

Auteur(s)	Devlin et al.
Année	2018
Langue(s)	104 langues aux plus gros Wikipédias
Taille	Normal (110M de paramètres) / Large (340M de paramètres)
Tâche(s) d'entraînement	MLM + prédiction de la phrase suivante
Corpus d'entraînement	Wikipédia des 104 langues (12500M mots) 20 Go
Segmentation (sous-mots)	WordPiece (vocabulaire de 110k sous-mots)

3.2.3 Réduction dimensionnelle (MLP)

Par souci de simplicité, seul le vecteur du premier sous-mot (que l'on note e_i) de chaque token (noté w_i) est récupéré et alimenté au module de réduction dimensionnelle. Ce module de réduction dimensionnelle est composé de quatre différents perceptrons multi-couches (Multi Layer Perceptron, MLP) et a pour but de réduire la dimension du vecteur afin d'alleger le modèle et d'éliminer du bruit ou de la redondance dans les vecteurs initiaux. Cela a le double bénéfice d'augmenter la vitesse de parsing en réduisant la complexité algorithmique du classifieur bi-affine ainsi que de réduire le sur-apprentissage du classifieur. Les quatre MLPs vont extraire une information différente en fonction de si le mot w_i en entrée est dépendant (*-d) ou gouverneur (*-h) et si la propriété à prédire est la gouvernance (arc-*) ou la relation syntaxique (rel-*).

$$h_i^{arc-h} = MLP^{arc-h}(e_i) \in \mathbb{R}^{k \times 1}$$

$$h_i^{arc-d} = MLP^{arc-d}(e_i) \in \mathbb{R}^{k \times 1}$$

$$h_i^{rel-h} = MLP^{rel-h}(e_i) \in \mathbb{R}^{k \times 1}$$

$$h_i^{rel-d} = MLP^{rel-d}(e_i) \in \mathbb{R}^{k \times 1}$$

Les représentations $h_1, ..., h_n$(des n mots de la séquence) sont ensuite concaténées pour former les quatre matrices H^{***-*} suivantes :

$$H^{arc-h} = (h_1^{arc-h}, ..., h_n^{arc-h}) \in \mathbb{R}^{k \times n}$$

$$H^{arc-d} = (h_1^{arc-d}, ..., h_n^{arc-d}) + 1 \in \mathbb{R}^{k \times n}$$

$$H^{rel-h} = (h_1^{arc-h}, ..., h_n^{rel-h}) \in \mathbb{R}^{k \times n}$$

$$H^{rel-h-d} = (h_1^{arc-d}, ..., h_n^{rel-d}) + 1 \in \mathbb{R}^{k \times n}$$

Pour chaque séquence, on va donc obtenir quatre matrices distinctes qui encodent des informations pertinentes différentes utilisées en entrée du classifieur bi-affine.

3.2.4 Classifieur bi-affine (CBA)

Pour produire la prédiction $\hat{y}_i^{arc}$ du gouverneur du mot w_i, on utilise le classifieur bi-affine suivant :

$$S_i^{arc} = H^{arc-h} W^{arc} h_i^{arc-dep} + H^{arc-h} + b^{T(arc)} \in \mathbb{R}^{n_{token}}$$

Le premier terme correspond à la probabilité que le mot j soit le gouverneur de i sachant les deux mots. Le deuxième terme correspond à la probabilité que le mot j soit gouverneur de i sachant seulement le mot j. Cela correspond à un biais sur la tendance des mots à être naturellement des gouverneurs en temps normal (par exemple, le biais d'un verbe sera souvent plus élevé que le biais d'un adverbe puisque les adverbes sont moins souvent des gouverneurs que les verbes).

Et donc pour les gouverneurs j potentiels de la phrase, celui qui maximise ce score s_{ij}^{arc} est le gouverneur le plus probable.:

$$\hat{y}_i^{arc} = argmax_{j \in [1,n]} s_{ij}^{arc}$$

Pour la prédiction de la fonction syntaxique avec le gouverneur du mot w_i, on prédit un vecteur de prédiction de sa relation potentielle syntaxique avec chaque mot w_j de la phrase. Voici la formule si l'on prend seulement la composante liée au gouverneur $\hat{y}_i^{arc}$ prédit plus tôt :

$$S_i^{rel} = h_{\hat{y}_i^{arc}}^{T(rel-head)} U^{rel} h_i^{rel-dep} + W^{rel} + (h_{\hat{y}_i^{arc}}^{T(rel-head)} + h_i^{T(rel-dep)}) + b^{rel} \in \mathbb{R}^{n_{token}}$$

Le premier terme de cette équation correspond à la probabilité d'une étiquette l à être la relation syntaxique associée à la dépendance de i par rapport à son gouverneur $\hat{y}_i$. Le deuxième terme représente la probabilité de la relation entre w_i et $\hat{y}_i$, quelle que soit la hiérarchie syntaxique entre les deux mots. Le troisième terme correspond au biais a priori d'avoir la relation syntaxique k comme étiquette.
Et donc pour les L étiquettes syntaxiques l possibles, celle maximisant ce score s_{il}^{rel} est la prédiction la plus probable :

$$\hat{y}_i^{rel} = argmax_{l \in L} s_{il}^{rel}$$

Les deux équations précédentes ne représentent qu'une dimension du calcul réel. En réalité, nous avons affaire à matrice de scores $S^{arc} \in \mathbb{R}^{n \times n}$ pour la prédiction des gouverneurs et $S^{rel} \in \mathbb{R}^{m \times n \times n}$ pour la prédiction des relations syntaxiques possibles associées à chaque couple w_i, w_j des tokens de la phrase.

3.2.5 Algorithme de reconstruction d'arbres

Pour reconstruire l'arbre de dépendances, on peut directement utiliser la matrice S^{arc} et y appliquer l'algorithme d'Edmond-Chu-Liu (ECL) afin d'obtenir l'arbre acyclique maximisant le score sur S^{arc}.

$$arc = ECL(S^{arc})$$

Les arbres reconstruits par l'algorithme ECL peuvent être non projectifs. Cela ne nous dérange pas car des constructions projectives sont présentes selon le formalisme d'annotation SUD du naija. Si le formalisme choisi interdit les constructions projectives non projectives, alors nous aurions dû nous tourner vers l'algorithme d'Eisner qui permet d'assurer la projectivité des arbres (pour une complexité algorithmique en $O(n^3)$ contrairement à ECL qui est en $O(n^2)$).

Une fois cet arbre reconstruit, il nous suffit de récupérer dans la matrice S^{rel} les composantes correspondantes aux liaisons syntaxiques de notre arbre.

$$rel = argmax(S^{arc}[index(S^{arc})])$$

Algorithme 1 : Algorithme de Chu-Liu Edmonds

Data : $G = (V, E)$ with the edge weight function $s : E \to \mathbb{R}$
begin
  Let $M = \{(x^*, x) : x \in V, x^* = argmax_{x'}(s(x, x'))\}$
  if G_m *has no cycle* **then**
    /* It is already an MST */
    return G_m
  else
    /* Find a cycle C in Gm */
    Let G_c be the subgraph of G excluding nodes in C
    Add a note c to G_c representing cycle C
    for $x \in V - C : \exists x' \in C/(x', x) \in E$ **do**
      Add edge (c, x) to G_c with $s(c, x) = max_{x' \in C}[s(x', x)]$
    end
    for $x \in V - C : \exists x' \in C/(x, x') \in E$ **do**
      Add edge (x, c) to G_c with
       $s(c, x) = max_{x' \in C}[s(x', x) - s(a(x'), x') + s(C)]$where $a(v)$ is the predecessor of v in C
    end
    Let $y =$ Chu-Liu-Edmonds(G_c, s)
    Find a vertex $x \in Csothat(x', x) \in y, (x'', x) \in C$
    return $y \cup C - \{(x'', x)\}$
  end
end

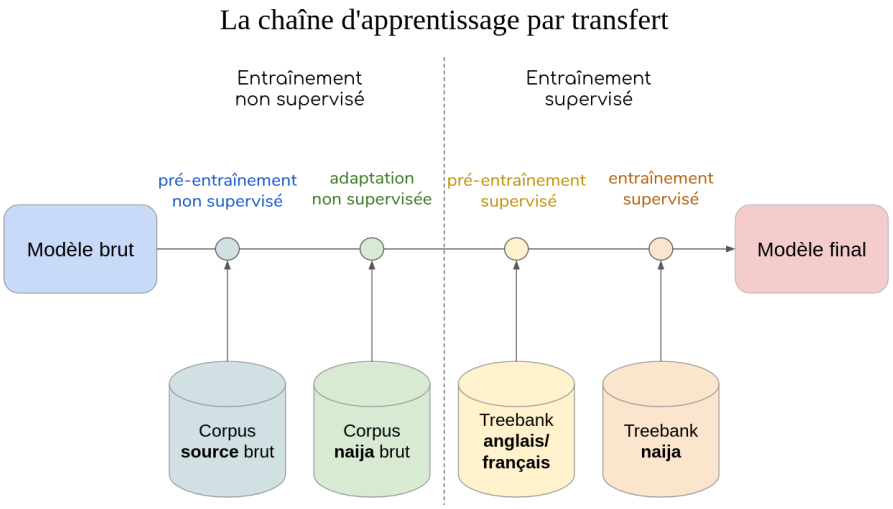

Figure 3.3: La chaîne d'apprentissage par transfert

3.2.6 Choix des modèles de transformers

3.3 Processus d'entraînement

Le processus entier d'entraînement d'un parseur syntaxique transformer + bi-affine peut comporter plusieurs étapes en fonction des différents pré-entraînements et adaptations effectués. Nous pouvons voir sur la figure 3.3 les quatre différents entraînements possibles lors de la chaîne d'apprentissage.

3.3.1 Pré-entraînement non supervisé

Dans un premier lieu, il est nécessaire de pré-entraîner les représentations cachées du transformer en effectuant une tâche d'apprentissage automatique non supervisée de modélisation de la langue (prédiction de mots masqués, prédiction du mot suivant, etc. . .). Nous montrons dans la section 4.1.4 que cette étape est nécessaire pour le bon apprentissage. Puisque nous n'avons pas les ressources computationnelles pour effectuer cet entraînement avec nos données du naija, nous avons utilisé et comparé des transformers dans la section 4.1 afin de choisir la configuration de langue(s) et d'entraînement la plus adaptée à notre tâche de parsing syntaxique du naija.

3.3.2 Adaptation non-supervisée

Avant de passer à des données annotées (labellisées), on peut effectuer d'autres adaptation/pré-entraînement non supervisées. Par exemple, on pourrait choisir d'adapter le modèle sur un corpus brut de texte de naija (ou de langue proche) ou bien sur des corpus bruts de textes oraux (quel que soir la langue). De cette manière, on peut commencer a tirer profit de grandes quantités de données non annotées pour commencer la spécialisation du modèle sur le type de données de

notre tâche cible (annotation syntaxique du naija oral). Lors de ce mémoire, nous n'avons pas effectué ce type d'adaptation/pré-entraînement.

3.3.3 Pré-entraînement supervisé

Ensuite, après avoir pré-entraîné de maniere non supervisée, plusieurs choix s'offrent à nous. On peut par exemple pré-entraîner de manière supervisée le modèle sur des données annotées en syntaxe de dépendance d'autres langues que le naija. En faisant ainsi, on prépare le modèle a réaliser la tâche d'annotation syntaxique dans le but d'augmenter les performances finales grâce à un transfert de connaissances entre les tâches. Puisqu'il existe des treebanks de l'anglais annotés en syntaxe de dépendance, nous avons mené dans la section 4.1 des expériences pour observer l'effet de ce transfert.

3.3.4 Adaptation supervisée

Enfin, une fois le transformer pré-entraîné, on connecte la sortie de celui-ci à l'architecture bi-affine de Dozat et Manning (2016). Contrairement aux trois entraînements présentés précédemment, cet entraînement n'est pas facultatif. Il est celui qui va permettre au modèle d'apprendre la distribution des données du corpus cible. Pour se faire, on effectue alors la rétro-propagation du gradient sur la tâche supervisée de parsing syntaxique afin d'optimiser les coefficients du module bi-affine ainsi que du transformer (si ce dernier n'est pas gelé).

3.3.5 Paramètres

Pour l'entraînement de nos modèles, nous utilisons l'algorithme d'Adam (Kingma and Ba 2014) avec un taux d'apprentissage [3] de 2×10^{-5}. Pour la structure du réseau bi-affine, nous conservons les paramètres optimaux donnés par Dozat et Manning (2016). Le MLP dédié à la réduction de dimension pour la prédiction des arcs est composé de deux couches cachées de 500 neurones avec un taux de dropout[4] de 0,3 et des fonctions d'activations RELU. Le MLP de la partie label est composé de deux couches cachées de 100 neurones avec un taux de dropout de 0,3 et des fonctions d'activations RELU.

Nous utilisons l'early-stopping (Morgan and Bourlard 1990) pour permettre d'arrêter l'entraînement

[3] Le taux d'apprentissage correspond à l'importance accordée au gradient d'erreur de la prédiction avec la valeur réelle. Plus ce taux sera important, plus les coefficients du modèle changeront après chaque rétro-propagation du gradient. Un taux d'apprentissage plus important n'est pas synonyme d'une convergence du modèle plus rapide puisque si les coefficients changent trop fortement entre deux rétro-propagations, il est possible de manquer des configurations locales intéressantes.

[4] Le *dropout* (ou *décrochage* en français) est une technique de régularisation pour éviter le sur-apprentissage du modèle. Lors de l'entraînement, certains neurones seront désactivés aléatoirement à chaque itération de rétro-propagation du gradient ce qui favorisera la redondance des expertises des neurones. Le dropout n'est pas activé lors de la phase d'inférence du modèle mais seulement lors de la phase d'entraînement

lorsque les performances du modèle n'évoluent plus sur les données d'évaluation après un certain nombre d'époques.

Le code source du projet est disponible sur GitHub[5]. Nous avons entraîné les modèles sur les serveurs de l'INRIA Paris au sein de l'équipe Almanach.

3.3.6 Scores d'évaluation

Pour juger de la qualité de nos modèles, nous avons utilisé plusieurs métriques d'évaluation que nous définissons ci-dessous.

Unlabelled Attached Score (UAS) : Terme employé pour désigner le score d'attachement non labellisé. Ce score permet d'évaluer la performance du parseur à trouver les bons gouverneurs des tokens. Il est calculé en calculant le ratio d'attachements syntaxiques corrects sur le nombre total d'attachements syntaxiques dans le corpus.

Labelled Unattached Score (LUS) : Terme employé pour désigner le score de labellisation (non attachée). Ce score permet d'évaluer la performance du parseur à trouver les bonnes fonctions syntaxiques entre ce token et son gouverneur (quel que soit ce dernier). Il est calculé en calculant le ratio de fonctions syntaxiques correctes sur le nombre total de fonctions syntaxiques dans le corpus.

Labelled Attached Score (LAS) : Terme employé pour désigner le score de labellisation attachée. Ce score permet d'évaluer la performance du parseur à trouver les bonnes fonctions syntaxiques ET les bons attachements entre les tokens et leurs gouverneurs. Il est calculé en calculant le nombre de fois ou l'attachement et la fonction syntaxique sont correctes et en divisant cela par le nombre total de tokens.

[5] `https://github.com/kirianguiller/BertForDeprel`

Chapitre 4

Expériences

Ce chapitre est dédié à la présentation et à l'analyse des expériences menées dans le but d'étudier l'influence de l'apprentissage par transfert sur les performances et la vitesse d'apprentissage d'un modèle d'analyse syntaxique du naija. Nous comparons dans la section 4.1 différents transformers pré-entraînés disponibles en fonction de certaines variables (langue de pré-entraînement, taille du modèle, tâche de pré-entraînement). Nous étudions en section 4.2 l'effet du gèle des coefficients du transformer sur l'entraînement de notre tâche cible. La section 4.3 étudie l'influence d'un pré-entraînement supervisé sur un treebank de l'anglais. Enfin, la section 4.4 examine l'impact de la taille du corpus cible sur les résultats finaux.

4.1 Influence du pré-entraînement non supervisé

Nous étudions dans une première partie l'influence du pré-entraînement non supervisée sur les performances de l'analyse syntaxique du naija lors de la phase d'adaptation. Pour se faire, nous avons utilisé des modèles disponibles publiquement sur la plateforme de partage de modèles Huggingface[1].

4.1.1 Langue(s) d'entraînement

Le premier facteur qui semble a priori le plus crucial pour le choix du modèle de langue (language model) est celui de la langue d'entraînement. Nous avons sélectionné, parmis les modèles BERT classiques, quatre configurations de langues différentes à savoir : BERT anglais, BERT chinois (mandarin), BERT multilingue et BERT français (FlauBERT). Pour le français, nous avons choisi FlauBERT (au profit de CamemBERT) puisque le modèle est basé sur l'architecture BERT (alors que CamemBERT est basé sur l'architecture RoBERTa).

Nous avons entraîné ces quatre modèles dans des conditions identiques (partitionnement du

[1] `https://huggingface.co/models`

dataset, taux d'apprentissage, batch size, etc.) et avons répété les entraînements 5 fois en changeant la graine aléatoire (random seed) du modèle. En répétant l'expérience, nous nous sommes assurés de réduire l'effet aléatoire de l'entraînement des modèles qui peut être important (cet effet aléatoire est surtout dû à l'approximation du gradient faite lors de la technique de descente du gradient stochastique ainsi qu'à la sélection itérative aléatoire des échantillons d'entraînements (batch)).

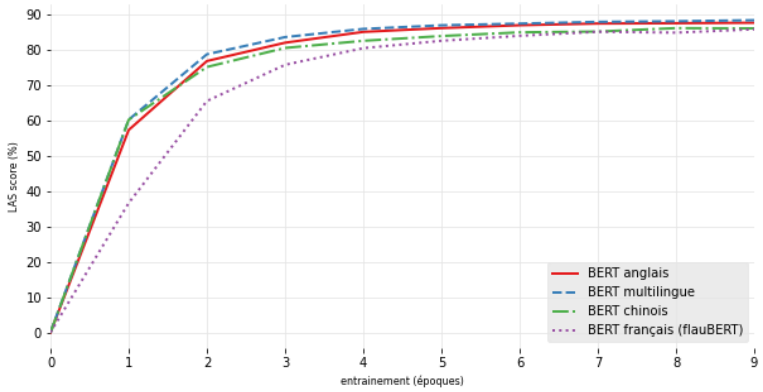

Figure 4.1: Évolution du score d'attachement labellisé (LAS) sur les 10 premières époques pour les configurations de langues suivantes : anglais; multilingue; chinois; français. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

modèle	POS	LUS	UAS	LAS	époques
BERT anglais	98.13	95.21	92.73	89.38	28.75
BERT multilingue	97.88	95.11	92.95	89.61	27.2
BERT chinois	96.74	94.24	91.76	87.83	20.0
BERT français (flauBERT)	97.68	94.93	92.13	88.45	28.2

Table 4.1: Performances atteintes pour les modèles anglais, multilingue, chinois et français. La colonne *époques* représente la valeur dès lors que le modèle ne progresse plus. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

Le tableau 4.1 nous présente les performances maximales atteintes des quatre modèles (moyennées sur les cinq graines aléatoires).
Nous pouvons voir que la catégorisation en parties du discours ainsi que la labellisation de la fonction syntaxique sont mieux réalisées par le modèle BERT anglais. Par contre, le modèle multilingue est plus performant pour la prédiction de l'attachement syntaxique. Au niveau des performances, les modèles anglais et multilingue sont plus proches entre eux qu'ils ne le sont avec les modèles chinois et français. Nous expliquons cela par le fait que l'anglais est la langue lexicalisante du naija et qu'il est la langue la plus représentée dans le corpus du modèle multilingue. Deux phénomènes peuvent être la raison qu'un modèle possède de meilleures capacités à transférer les connaissances :

- Le **recouvrement lexical** entre les langues sources et cibles influe

- Le **recouvrement structurel** entre les langues sources et cibles influe.

	Tokenisations par mot	
	moyenne	écart-type
BERT anglais	1.18	0.24
BERT multilingue	1.17	0.23
BERT chinois	1.46	0.33
BERT français (FlauBERT)	1.35	0.33

Table 4.2: Moyenne et écart-type de la tokenisation des mots en fonction de la langue

Nous nous sommes d'abord intéressé au recouvrement lexical. Le tableau 4.2 montre le nombre moyen de segmentations par mot des segmenteurs en sous-mots des quatre langues. Les segmenteurs utilisés sont spécifiques à chaque modèle BERT et sont utilisés en amont de celui-ci. Avant de pouvoir commencer l'entraînement d'un modèle BERT, il faut d'abord entraîner un segmenteur sur le corpus brut d'entraînement. Ce segmenteur est composé d'un algorithme qui détecte les N suites de caractères les plus fréquentes dans le corpus d'entraînement avec N étant la taille du vocabulaire du tokeniseur. Ces N sous-mots disposent alors d'une entrée dans le tokeniseur et sont associés à un plongement lexical qui sert de vecteur d'entrée du modèle BERT. Le critère de segmentation est seulement basé sur la fréquence d'apparition des chaînes de caractères et non pas sur la morphologie de la langue !

L'exemple de phrase tokenisée par le tokeniseur de CamemBERT illustre ceci :

Les pinguins glissent sur la banquises

Les pin guin s glisse nt sur la ban qui s e

Ici (table 4.2) les modèles anglais et multilingue ont la même moyenne de segmentation du naija (1,17 séparation en moyenne par mot). Cela s'explique par le fait que le modèle multilingue possède un vocabulaire plus grand (110k sous-mots) que l'anglais (42k). Ensuite, nous pouvons observer que le modèle FlauBERT possède un plus grand recouvrement du naija que le BERT chinois puisque ce dernier segmente les mots du naija en plus de sous-mots (1,46 sous-mots par token en moyenne) que FlauBERT (1,35 sous-mots/token). Les corpus anglais et multilingue ont donc des meilleurs recouvrement que le corpus français qui lui-même a un meilleur recouvrement que corpus chinois. Nous pourrions donc penser que le recouvrement lexical est le phénomène qui est responsable du bon transfert.

Cependant, (Karthikeyan et al. 2019) affirment que le recouvrement lexical entre deux langues n'apporte quasiment rien dans le gain de performance dû à l'apprentissage par transfert. Selon eux, la similarité structurelle entre les langues ainsi que la profondeur du réseau (c.-à-d. nombre de couches) sont des facteurs plus importants pour la capacité à transférer des connaissances.

			XNLI		NER
B-BERT	**Train**	**Test**	**Accuracy**	**Word-piece Contribution**	**F_1-Score**
en-es	en	es	72.3	1.4	61.9 (±0.8)
enfake-es	enfake		70.9		62.6 (±1.6)
en-hi	en	hi	60.1	0.5	61.6 (±0.7)
enfake-hi	enfake		59.6		62.9 (±0.7)
en-ru	en	ru	66.4	0.7	57.1* (±0.9)
enfake-ru	enfake		65.7		54.2 (±0.7)
en-enfake	enfake	enfake	78.0	0.5	78.9*(±0.7)
en-enfake	enfake	en	77.5		76.6(±0.8)

Table 1: **The Effect of Word-piece Overlap and of Structural Similarity** For different pairs of B-BERT languages, and for two tasks (XNLI and NER), we show the contribution of word-pieces to the success of the model. In every two consecutive rows, we show results for a pair (e.g., English-Spanish) and then for the corresponding pair after mapping English to a disjoint set of word-pieces. The gap between the performance in each group of two rows indicates the loss due to completely eliminating the word-piece contribution. We add an asterisk to the number for NER when the results are statistically significant at the 0.05 level.

Figure 4.2: Effet du recouvrement lexical et de la similarité structurelle sur le fine-tuning des transformers. Figure issue de (Karthikeyan et al. 2019)

Pour prouver ceci, les auteurs pré-entraînent des modèles BERT sur des corpus dont chaque caractère unicode est indenté d'une valeur constante suffisamment grande afin de s'assurer à ce qu'il n'y ait plus de recouvrement lexical entre ce nouveau "faux-anglais" (dénomination des auteurs) et les autres langues de l'expérience (russe, espagnol et hindi) dans le tokenizer. Ils évaluent ensuite les performances des modèles de l'anglais et du faux-anglais sur différentes tâches annexes dans les trois langues citées et observent effectivement que la performance finale varie plus en fonction de la langue cible (espagnol, hindi ou russe) que de la langue source (anglais ou faux-anglais). Le tableau 4.2 montre ces résultats et l'on peut voir que l'absence de recouvrement lexical pour le modèle faux-anglais n'est pas très influent.
Si nous nous fions à ces résultats, alors le recouvrement structurel entre deux langues est le facteur le plus important pour un bon transfert de connaissances. Cela est aussi en accord avec nos résultats puisque le modèle anglais performe un peu mieux que les modèles chinois et français. Les mots de la phrase ont sensiblement le même arrangement en anglais et en naija et cela pourrait donc aider le transformers lors de l'adaptation. Quant au modèle multilingue, il performerait mieux que les autres grâce au fait qu'étant entraîné sur 104 langues, certaines constructions présentes en naija mais absentes en anglais seraient apprises car présentes dans d'autres langues. Le modèle n'aurait donc qu'à ajuster des attentions de ses premières couches pour pouvoir retomber sur des connexions pré-construites des dernières couches.

Nous nous sommes ensuite intéressés aux vitesses d'apprentissages des différents modèles. Le graphique 4.1 nous montre l'évolution de l'entraînement des quatre configurations différentes. Nous avons, après chaque époque d'entraînement, évalué les modèles sur les données d'évaluation. On peut voir que les trois modèles BERT s'entraînent sensiblement plus rapidement que le modèle FlauBERT français. Nous pensons que cet écart d'apprentissage est plus lié à un problème technique du modèle que la dissimilarité linguistique du français avec le naija puisque le modèle du chinois, une langue plus éloignée du naija que le français, s'entraîne bien plus vite que le modèle du français.

Un autre point intéressant mis en valeur par ces résultats est que le modèle multilingue s'entraîne plus rapidement que le modèle anglais. La figure 4.3 montre l'avance de l'apprentissage du modèle multilingue sur le modèle anglais au niveau du LAS. Nous pouvons observer que l'écart est de 3% à la première époque et qu'il diminue par la suite. On a donc un modèle multilingue capable d'apprendre légèrement plus rapidement la résolution de la syntaxe que le modèle anglais.

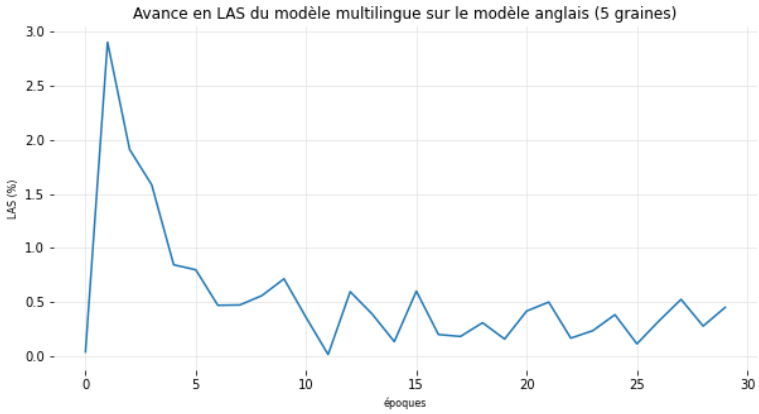

Figure 4.3: Avance sur le LAS score du modèle multilingue par rapport au modèle anglais. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

Les modèles multilingue et anglais sont capables d'un meilleur transfert de connaissances et sont donc ceux que nous allons retenir par la suite (nous utiliserons néanmoins à nouveau, pour certaines expériences, des modèles autres que le multilingue et l'anglais dans le but de faire des comparaisons précises).

4.1.2 Tâche(s) d'entraînement

Le modèle BERT classique est entraîné sur deux tâches qui sont la prédiction de mots masqués ainsi que la prédiction de la phrase suivante. Les auteurs de RoBERTa (Y. Liu et al. 2019) affirment avoir, pour un nombre de paramètres égal, amélioré les résultats de BERT en enlevant la tâche de prédiction de la phrase suivante et en augmentant le nombre de données d'entraînement. Nous montrons ici que cela n'a pas réellement d'impact sur la tâche de parsing syntaxique du naija.
Nous voyons sur la figure (4.4) que le modèle RoBERTa n'est pas plus performant que le modèle BERT. L'entraînement du modèle RoBERTa est plus lent dans les 5 premières époques. Le tableau 4.3 résume les performances maximales atteintes par les deux modèles.
Les résultats montrent que les deux modèles atteignent sensiblement les mêmes performances. Même si le modèle RoBERTa est plus lent sur les premières époques, il atteint finalement une performance légèrement supérieure au modèle BERT pour la précision de l'attachement syntaxique. Au vu de la très faible différence, nous en concluons que le choix entre ces deux configurations d'entraînement n'a pas de grande influence vis-à-vis de notre problématique cible. Nous n'avons pas réellement d'hypothèses pour expliquer celà, d'autant plus que nous n'avons pas moyen d'isoler les deux sources de variations majeures entre ces deux modèles (tâches d'entraînement

et taille du corpus d'entraînement).

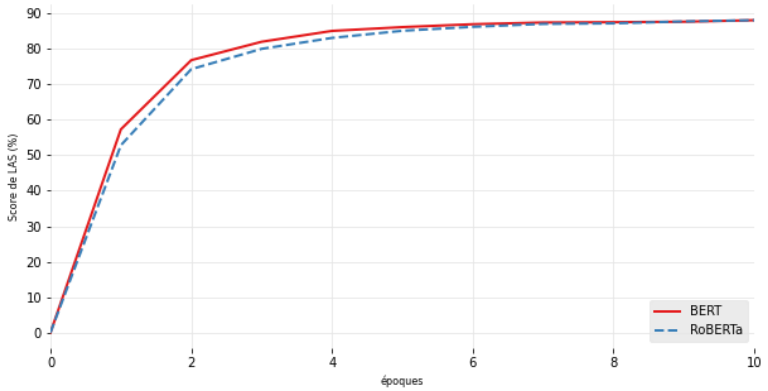

Figure 4.4: Évolution du score d'attachement labellisé (LAS) sur les 10 premières époques pour les modèles suivants : BERT anglais et RoBERTa anglais

modèle	POS	LUS	UAS	LAS	époques
BERT	98.13	95.21	92.73	89.38	28.75
RoBERTa	98.1	95.08	92.9	89.46	32.6

Table 4.3: Performances atteintes pour les modèles suivants : BERT anglais et RoBERTa anglais. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

4.1.3 Taille du modèle

Pour une architecture de transformer donnée, la taille de celui-ci a une incidence importante sur sa capacité à se construire une représentation du langage et à mémoriser les données. Les auteurs de BERT, RoBERTa et ALBERT convergent d'ailleurs sur cet aspect et confirment tous que la performance du modèle s'améliore avec la taille de celui-ci. Récemment, (Brown et al. 2020) d'OpenAI ont publié un article relatant des performances de leur nouveau modèle GPT-3 en fonction de la taille. Le modèle final, de 175 milliards de paramètres (un nombre jamais atteint auparavant), possède une très bonne représentation de la langue (majoritairement anglaise) et est capable d'apprendre de nouvelles tâches avec pas ou très peu d'exemples (resp. communément appelées *zero-shot learning* et *few-shot learning*)

Cependant, cette solution d'utiliser des modèles plus gros pour obtenir de meilleurs résultats n'est pas forcément raisonnée. Le gain obtenu en performance doit être comparé avec le coût supplémentaire (aussi bien temporel, économique et surtout écologique) à fournir lors de l'entraînement ainsi qu'en production. Il peut être tentant de faire un modèle deux fois plus grand afin de dépasser l'état de l'art pour le "simple" coût d'un temps d'entraînement deux fois plus long, mais cela peut avoir des répercussions massives de consommations en aval lors de l'utilisation de ces modèles en production. Une recherche récente (Thompson et al. 2020)

montre que les techniques de deep learning vont bientôt arriver à un stade ou les ressources computationnelles, énergétiques et financières seront trop importantes et freineront la vitesse des avancées technologiques.

Le nombre de paramètres d'un transformers BERT classique dépend des variables suivantes :

- du nombre de couches dont il dispose.
- de la dimension de la représentation vectorielle.
- du nombre de têtes d'attention.

Nous avons entraîné, pour trois tailles de modèles différentes (ALBERT, BERT-base et BERT-large), des modèles dont voici en fig 4.5 les courbes d'apprentissages.
On peut rapidement observer que le modèle le plus large se détache des autres modèles en termes de rapidité d'adaptation [2]. Nous pensons que cela est lié au plus grand nombre de mécanismes d'attention présent dans le modèle *large* qui fait qu'il possède plus de connexions pré-entraînées modélisant différentes régularités statistiques du corpus initial (ici le Wikipédia anglais ainsi que des romans) prêtent à être ajustées légèrement pour s'adapter sur un autre domaine.

Le tableau 4.4 résume les scores atteints par les parseurs. Les modèles BERT basique et large ont des précisions très proches pour les prédictions de la catégorie du discours et de la fonction syntaxique (resp. 0.01% et 0.01% d'écart) et ont des précisions 50 fois plus éloignées pour la prédiction de l'attachement labellisé (0,5% d'écart). Cet écart est significatif et peut indiquer une différence de difficulté de résolution entre ces deux tâches.

modèle	POS	LUS	UAS	LAS	époques
small (ALBERT)	97.72	94.51	91.44	87.65	24.75
medium (BERT-base)	98.13	95.21	92.73	89.38	28.75
large (BERT-large)	98.11	95.13	93.13	89.74	26.2

Table 4.4: Performances atteintes pour les modèles configurations de taille suivantes : small (AlBERT); medium (BERT-base); large (BERT-large). (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

La figure 4.6 présente l'évolution du LUS (précision de la labellisation de la fonction syntaxique) et la figure 4.7 présente l'évolution du UAS (précision de l'attachement du gouverneur syntaxique). On observe deux choses : Le choix du modèle est très peu influent sur la précision de la prédiction de la fonction syntaxique et est plus influent sur la précision de l'attachement syntaxique. La précision de la prédiction de la fonction syntaxique atteint 90% dès la première

[2]par rapidité d'adaptation, nous entendons le nombre d'itérations d'entraînement nécessaires pour fitter les données et non pas le temps nécessaire en secondes pour fitter les données. Cela peut paraître contre-intuitif puisqu'un modèle plus large implique d'avoir plus de paramètres à optimiser par rétro-propagation du gradient et on pourrait donc émettre comme a priori qu'il faudrait plus d'iterations d'entraînement pour fitter aux données (nous avons utilisé le même taux d'apprentissage ainsi que le même optimiseur).

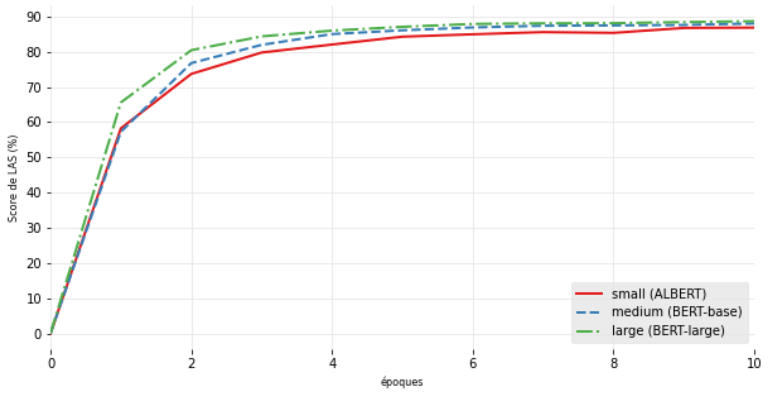

Figure 4.5: Évolution du score d'attachement labellisé (LAS) sur les 10 premières époques pour les configurations de taille suivantes : small (AlBERT); medium (BERT-base); large (BERT-large). (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

époque (quel que soit le modèle) alors que la précision de la prédiction de l'attachement atteint ces même 90% après 4 époques pour le modèle large, 5 époques pour le modèle basique et 9 époques pour le modèle FlauBERT.

Il y a donc une profonde asymétrie entre les deux tâches élémentaires du parsing syntaxique qui ne disposent pas des mêmes courbes d'apprentissage. Selon nous, cela pourrait s'expliquer par le fait que les modèles de la langues (de type transformer) possèdent, pour une séquence de tokens, une représentation des interactions entre tous ces tokens et il suffirait alors d'ajuster certains paramètres pour faire concorder ces représentations avec les fonctions syntaxiques de la problématique. En revanche, la structure arborée telle que formalisée par la syntaxe de dépendance ne serait pas directement apprise lors du pré-entraînement et une adaptation plus longue serait nécessaire. Formulé autrement, nous pensons qu'un transformer possède une bonne connaissance sur les relations d'un mot avec tous les autres mais pas forcement une bonne connaissance sur lequel de ces mots est son gouverneur syntaxique de dépendance.

À l'avenir, nous voudrions faire une expérience où l'on entraînerait une architecture BERT pour les deux tâches séparément (transformer + bi-affine-head / transformer+bi-affine-deprel). Après entraînement, nous voudrions comparer l'écart entre les poids initiaux et finaux afin d'observer s'il est plus important pour le modèle qui doit reconstruire l'arbre syntaxique. Si tel est le cas, alors peut-être qu'une information plus grande sur les fonctions syntaxiques est apprise lors du pré-entraînement que sur l'arbre syntaxique.

4.1.4 Baseline : initialisation aléatoire des modèles

Les auteurs de l'architecture RoBERTa (Y. Liu et al. 2019) affirment avoir observé lors de leurs expérimentations que le modèle BERT était sous-entraîné et qu'il était possible d'améliorer ses performances sur les tâches en aval en l'entraînant plus longtemps sur un volume de données

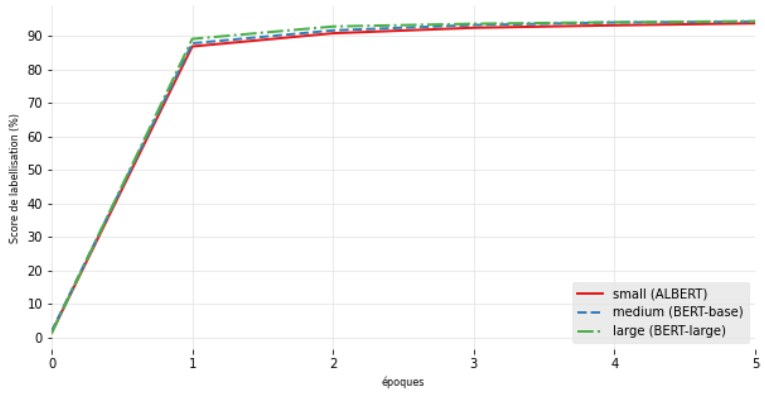

Figure 4.6: Évolution du score de labellisation (LUS) sur les 5 premières époques pour les configurations de taille suivantes : small (AlBERT); medium (BERT-base); large (BERT-large)

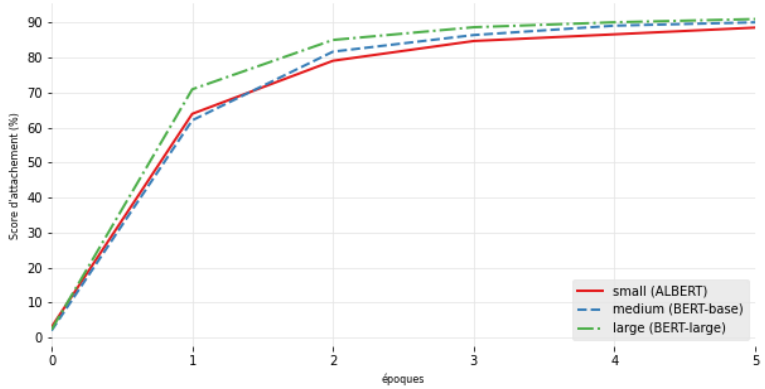

Figure 4.7: Évolution du score d'attachement non labellisé (UAS) sur les 5 premières époques pour les configurations de taille suivantes : small (AlBERT); medium (BERT-base); large (BERT-large)

plus important. Les auteurs de BERT ont d'ailleurs par la suite ré-entraîné les modèles BERT plus longtemps pour pallier ce manque.

Nous n'avons pas réellement la possibilité de faire une expérimentation où l'on contrôle le temps de pré-entraînement et nous partons du principe que les différents modèles, quelque soit leurs architectures, sont entraînés suffisamment longtemps pour que le potentiel d'amélioration restant soit négligeable.

Par conséquent, nous avons décidé à la place de faire une expérience contrôle en réinitialisant aléatoirement les coefficients des neurones du transformer pour observer les résultats. En faisant ceci, nous souhaitons confirmer que, pour notre tâche d'analyse syntaxique du naija, partir d'un modèle pré-entraîné apporte bien des connaissances par rapport à un modèle aléatoire.

La figure 4.8 compare l'entraînement d'un modèle pré-entraîné normalement sur l'anglais de manière non supervisée (courbe rouge) avec deux modèles initialisés aléatoirement (en bleu un modèle anglais et en vert un modèle chinois)[3]. Nous observons de ces entraînements que les versions réinitialisées aléatoirement n'apprennent pas aussi bien que les versions pré-entraînées 4.5 et plafonnent aux alentours de 80% de score d'attachement labellisé même pour un très grand nombre d'époques (le graphique ne reporte que les 30 premières époques, mais les valeurs rapportées dans le tableau pour les modèles réinitialisés correspondent à la cinquantième époque).

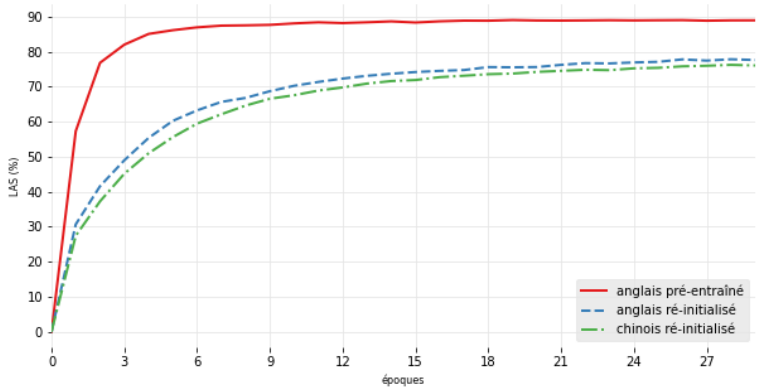

Figure 4.8: Evolution du LAS lors de l'entraînement du modèle anglais avec et sans ré-initialisation aléatoire et du modèle chinois avec ré-initialisation aléatoire des coefficients. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

modèle	POS	LUS	UAS	LAS	époques
anglais pré-entraîné	98.13	95.21	92.73	89.38	28.75
anglais ré-initialisé	95.32	91.92	84.38	79.86	48.6
chinois ré-initialisé	94.01	91.22	82.99	78.27	48.8

Table 4.5: Performances atteintes pour le modèle anglais avec et sans ré-initialisation aléatoire et pour le modèle chinois avec ré-initialisation aléatoire des coefficients. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

Nous pouvons faire une autre observation qui est que parmi les deux modèles réinitialisés chinois et anglais, le modèle anglais est celui qui apprend le mieux. Cela peut s'expliquer par le recouvrement lexical plus important du vocabulaire du tokeniseur anglais vis-à-vis du tokeniseur chinois. Pour en avoir le cœur net, nous pourrions à l'avenir entraîner différents modèles disposant de vocabulaire plus ou moins couvrant afin d'observer s'il y a une réelle corrélation.

Au vu de ces résultats, nous confirmons l'hypothèse que le pré-entraînement du modèle de langue est recommandé si l'on veut espérer atteindre des performances élevées.

[3]Bien que les modèles soient réinitialisés aléatoirement, nous les considérons toujours comme des modèles "anglais" ou "chinois" puisque le vocabulaire du segmenteur en mot est lui toujours basé sur les formes apprises lors de leurs entraînements respectifs.

4.1.5 Conclusion de l'influence du pré-entraînement non supervisé

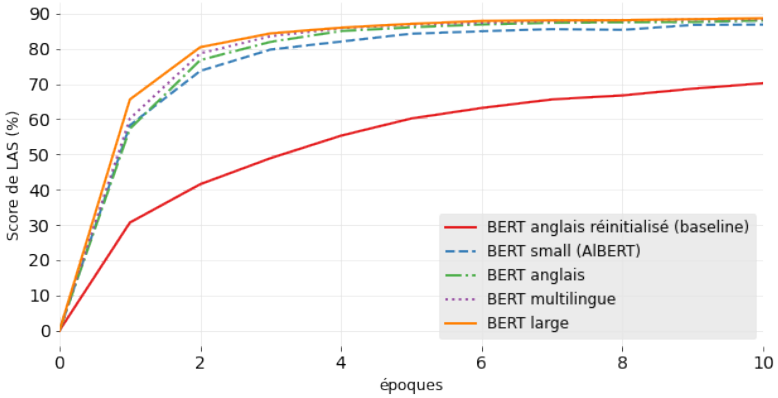

Figure 4.9: Influence de l'entraînement non-supervisé : résumé - Évolution du LAS (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

modèle	POS	LUS	UAS	LAS	époques
BERT anglais réinitialisé (baseline)	95.32	91.92	84.38	79.86	48.6
BERT small (AlBERT)	97.72	94.51	91.44	87.65	24.75
BERT anglais	98.13	95.21	92.73	89.38	28.75
BERT multilingue	97.88	95.11	92.95	89.61	27.2
BERT large	98.11	95.13	93.13	89.74	26.2

Table 4.6: Influence de l'entraînement non-supervisé : résumé - Performances atteintes par les différents modèles. (Schéma d'annotation : SUD; Corpus : Naija SynCor Treebank)

On a donc vu lors de ces expériences que le choix du modèle pré-entraîné est important. En premier lieu, nous pouvons observer que la totalité des modèles pré-entraînés de manière non supervisée sur des corpus bruts atteignent des performances largement supérieures au modèle non pré-entraîné (réinitialisé aléatoirement). Ensuite, si l'on désire améliorer encore les performances, choisir un modèle pré-entraîné sur un corpus d'une langue proche à celle de la langue cible assure de pouvoir effectuer un transfert de connaissance qui aidera à améliorer les performances finales. Cette similarité doit être principalement structurelle et/ou lexicale (nous n'avons pu différencier exactement la différence entre les apports de connaissances dû à ces deux similarités). Dans notre cas, bien que l'anglais soit très proche du naija, nous pouvons observer une légère amélioration des performances lors de l'utilisation d'un modèle multilingue. Les modèles multilingues apprennent plus vite et atteignent un seuil plus haut de performances. Nous pensons donc que, dans le cas d'un éloignement plus prononcé entre l'anglais et une langue cible, le modèle multilingue devrait se distinguer davantage.

Plus un modèle contient de paramètres, meilleure sera sa connaissance de la langue et plus performants seront ses résultats. Cependant, les modèles de plus grandes tailles sont aussi plus longs à entraîner et plus gourmands en ressources. Si les ressources computationnelles ne sont

pas limitées, alors il est possible de prendre le modèle le plus gros possible. Sinon, se contenter d'un modèle basique de 100M de paramètres ou même d'un modèle léger ou distillé [4] (ALBERT, DistilBERT) peut s'avérer être un bon choix.

4.2 Fine-tuning VS Features extraction

Cette section est dédiée à l'étude de l'effet du gèle des neurones du transformer lors de l'adaptation supervisée sur le corpus cible. Pour rappel, il existe deux approches différentes lorsque l'on désire faire de l'apprentissage par transfert séquentiel : l'ajustement (fine-tuning) ainsi que l'extraction de caractéristiques (feature extraction). Nous avons défini ces notions dans la section 2.4.4 et nous vous invitons à vous y référer si besoin.

Nous avons entraîné des modèles BERT anglais et multilingue avec des neurones "gelés" afin de comparer ces résultats aux entraînements de modèles normaux effectués dans la section précédente.

Les figures 4.10 et 4.11 montrent les évolutions des précisions du score d'attachement syntaxique et de la fonction syntaxique pour les approches de fine-tuning (lignes continues) et de features extraction (pointillés). Les modèles multilingues sont en rouge et les modèles anglais en bleu.

La première observation évidente est que les modèles gelés ont un apprentissage beaucoup plus lent que ce soit pour le score d'attachement ou le score de labellisation. Ceci montre donc l'importance de faire du fine-tuning plutôt que de l'extraction de caractéristiques dans notre cas.

On peut voir que l'impact de ce gel est beaucoup plus grand sur le score d'attachement syntaxique que sur le score d'étiquetage et cela recoupe les résultats obtenus dans la section 4.1.4. En effet, nous émettons l'hypothèse que lors du pré-entraînement, le modèle apprend de l'information recouvrant en plus grande partie avec notre problème de catégorisation syntaxique que de reconstruction de l'arbre de dépendance. Cela se vérifie ici puisque les deux tâches sont réalisées grâce à deux classifieurs bi-affines (voir 3.2.4) distincts possédant le même nombre de paramètres et ayant comme entrées les mêmes vecteurs issus de la dernière couche de BERT. Cependant, malgré cette similarité structurelle, le parseur dédié à la tâche de catégorisation de la fonction syntaxique apprend plus vite et atteint de meilleures performances.

Le graphique 4.12 nous montre l'évolution des modèles pour les 50 premières époques. Cette expérience nous apporte une information intéressante concernant la vitesse d'apprentissage du BERT anglais vis-à-vis du BERT multilingue. On peut voir que le modèle anglais apprend plus vite que son équivalent multilingue alors que pour rappel, le modèle multilingue est plus performant et apprend plus vite que le modèle anglais lorsque les coefficients ne sont pas gelés. Cela pourrait s'expliquer par le fait que le modèle anglais possède originellement une meilleure représentation correcte du lexique du naija et que le fait de geler les coefficients avantage le

[4] les modèles distillés sont des modèles dont seulement une partie de la représentation interne du modèle est conservée (pour faire les prédictions, l'ajustement, etc.

modèle de la langue la plus proche. Inversement, le modèle multilingue posséderait un meilleur potentiel d'apprentissage si l'on autorise l'optimisation des poids dû au fait que l'ajustement des premières couches permettrait de lier les entrées aux structures d'attention complexes présentes dans les dernières couches. Cela confirmerait que le corpus multilingue possède un meilleur recouvrement structurel avec le naija et que le corpus anglais possède un meilleur recouvrement lexical.

modèle	POS	LUS	UAS	LAS	époques
BERT multilingue	97.88	95.11	92.95	89.61	27.2
BERT multilingue (gelé)	91.39	90.73	72.67	68.11	49.8
BERT anglais	98.1	95.28	92.68	89.37	18.0
BERT anglais (gelé)	92.38	91.37	76.04	71.69	50.0

Table 4.7: Performances atteintes pour les configurations avec et sans figement des coefficients des modèle BERT anglais et multilingue

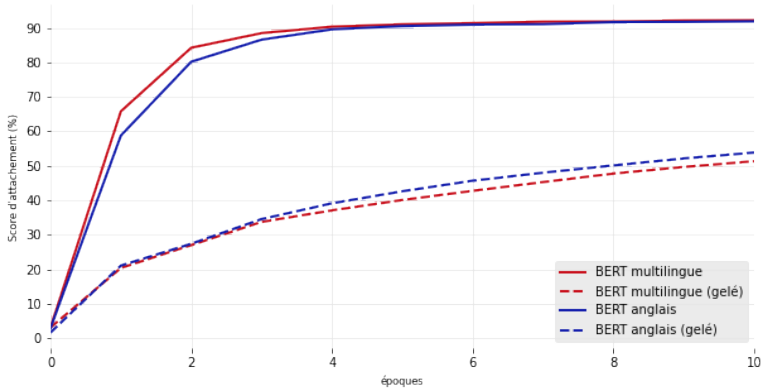

Figure 4.10: Évolution du score de labellisation non attachée (UAS) sur les 10 premières époques pour les configurations avec et sans figement des coefficients des modèle BERT anglais et multilingue

4.3 Influence du pré-entraînement supervisé

Cette section étudie l'influence d'un pré-entraînement supervisé de notre modèle sur des corpus d'autres langues. Ceci contraste avec la section 4.1 qui étudiait l'influence du pré-entraînement non supervisé du modèle.

Pour cela, nous avons pré-entraîné des modèles BERT multilingues dans plusieurs configurations différentes (treebank anglais; treebank anglais partiel; treebank du français oral) que nous avons ensuite adapté sur le treebank du naija.

Nous disposons de 8 treebanks de l'anglais qui possèdent des styles variés. Nous avons aggloméré ces treebanks en un seul que nous nommons "all_english". Ce treebank de 24000 phrases est

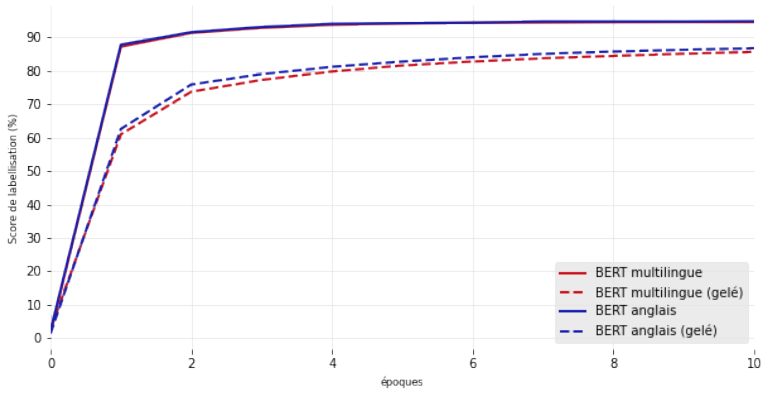

Figure 4.11: Évolution du score d'attachement non labellisé (LUS) sur les 10 premières époques pour les configurations avec et sans figement des coefficients des modèle BERT anglais et multilingue

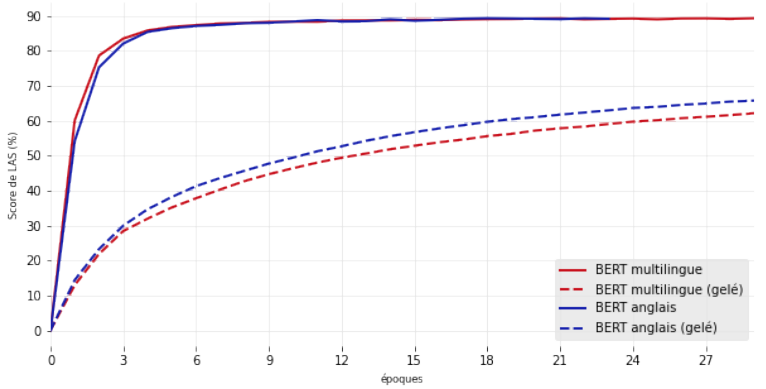

Figure 4.12: Évolution du score d'attachement labellisé (LAS) sur les 30 premières époques pour les configurations avec et sans figement des coefficients des modèle BERT anglais et multilingue

composé de textes écrits seulement et n'a donc aucune information sur la langue orale comme pour notre corpus du naija.

Nous disposons d'un treebank du français oral "french_spoken" issu du projet ANR Rhapsodie (Lacheret et al., 2014) mené par MoDyCo. Ce corpus est composé de 2000 phrases.

Enfin, pour pouvoir comparer les pré-entraînements provenant de l'anglais et du français de manière plus égale, nous avons sélectionné 2000 phrases du corpus total pour former un sous-corpus : "partial_english".

Un pré-entraînement supervisé est effectué respectivement sur ces trois corpus et est arrêté au bout de 10 époques sans améliorations. Le score d'attachement labellisé atteint est de 89,74% sur le french_spoken, 90,08% pour le all_english et 89,71% pour le partial_english.

Les figures 4.13, 4.14 et 4.15 montrent l'évolution des performances (resp. UAS, LUS et LAS)

pour ces trois configurations ainsi que pour le modèle non pré-entraîné. Les informations essentielles sont inscrites dans le tableau 4.8.

On observe dans cette configuration une très grande amélioration de la vitesse d'entraînement du système sur les premières époques lorsque les modèles sont pré-entraînés sur l'anglais ou le français. Les modèles pré-entraînés ont aussi une capacité de transfert en zero-shot non négligeable (entre 20 et 40% de LAS, sans compter la ponctuation). Nous voyons aussi qu'il faut un nombre moins grand d'époques (4 en moyenne) pour ajuster le modèle pré-entraîné sur le corpus anglais total par rapport au modèle de base. Bizarrement, le modèle pré-entraîné sur le corpus partiel s'adapte plus vite que celui sur le corpus total et nous n'avons pas d'explications pour ce phénomène.

Il est intéressant de remarquer que le modèle pré-entraîné sur le corpus *french spoken* possède des performances quasi-égales aux modèles pré-entraînes sur l'anglais. Ceci peut s'explique par la méthodologie d'annotation similaire sur les deux corpus French Spoken et NaijaSynCor (le découpage en phrase est effectué de la même manière) et les deux langues possèdent un grand nombre de dislocations.

Nous pouvons donc conclure qu'un transfert de connaissances syntactiques a lieu entre les corpus du français et de l'anglais vers le naija. Plus les données de pré-entraînement sont nombreuses, plus la vitesse de fitting du modèle lors de l'adaptation est rapide et plus la performance est élevée.

modèle	POS	LUS	UAS	LAS	époques
all english treebank	98.11	95.3	93.37	90.08	23.2
partial english treebank	98.12	95.23	92.99	89.71	26.8
french spoken	98.0	95.21	93.03	89.74	30.0
no pretrain	97.88	95.11	92.95	89.61	27.2

Table 4.8: Performances atteintes pour les modèles entraînés (moyennés sur 5 seeds) sur les trois différents corpus (all_english, partial_english, french_treebank) ainsi que le modèle non pré-entraîné

4.4 Influence de la taille du corpus d'adaptation

Nous avons vu dans les parties précédentes que plusieurs processus de transfert de connaissances ont lieu et que nous pouvons en bénéficier afin d'améliorer la performance et la vitesse d'apprentissage de notre analyseur syntaxique. Il faut garder à l'esprit que les expériences menées jusqu'à présent se basent sur plus de 6000 phrases annotées syntaxiquement du naija. Puisque nous voulons que nos travaux soient aussi utiles pour des projets de création de treebank, nous proposons de faire une expérience où le but est d'observer l'influence qu'a l'utilisation d'un modèle pré-entraîné pour des petits corpus.

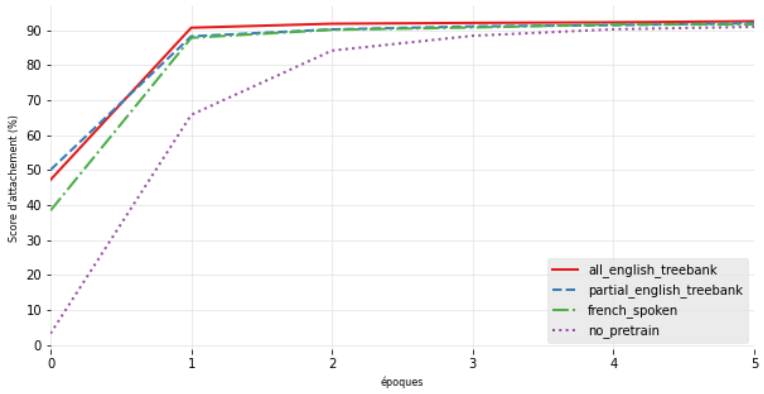

Figure 4.13: Évolution du score d'attachement non labellisé (UAS) sur les 10 premières époques pour les modèles entraînés sur les trois différents corpus (all_english, partial_english, french_-treebank) ainsi que le modèle non pré-entraîné

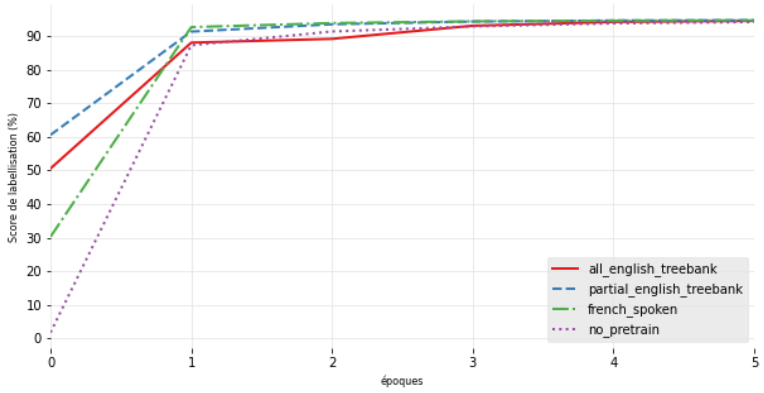

Figure 4.14: Évolution du score de labellisation non attachée (LUS) sur les 10 premières époques pour les modèles entraînés sur les trois différents corpus (all_english, partial_english, french_-treebank chinois) ainsi que le modèle non pré-entraîné

Pour cela, nous avons entraîné des modèles sur deux axes de variations :

- La taille du corpus d'entraînement (1/10/100/1000/5000 phrases)
- Le pré-entraînement du modèle (anglais/aucun)

Pour l'évaluation, nous avons utilisé 2200 phrases mises de côté. Le corpus d'évaluation est assez conséquent afin d'être sûr d'observer au mieux la généralisation des modèles. Nous avons fait en sorte que les corpus d'entraînement et d'évaluation ne partagent pas des documents en commun afin de ne pas avantager les modèles s'entraînant sur de grands échantillons (1000 et 5000 phrases) qui auraient alors plus de chance de s'entraîner sur du vocabulaire présent dans les corpus d'évaluation.

Pour pouvoir comparer les vitesses d'apprentissage, nous avons fixé la taille des batchs à 1 (la

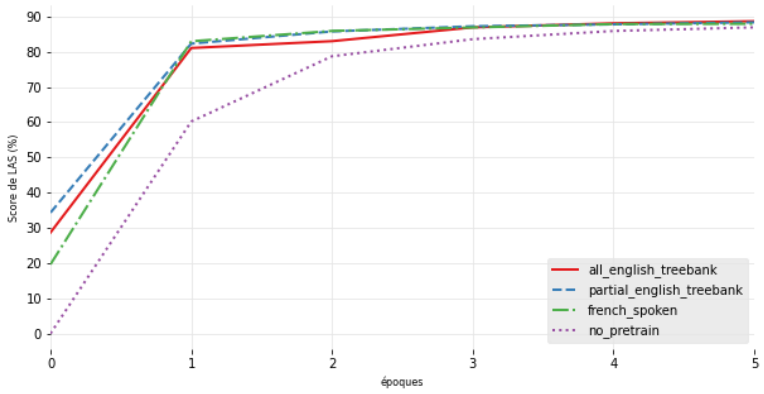

Figure 4.15: Évolution du score d'attachement labellisé (LAS) sur les 10 premières époques pour les modèles entraînés sur les trois différents corpus (all_english, partial_english, french_treebank chinois) ainsi que le modèle non pré-entraîné

retro-propagation est calculée après chaque phrase) et comparé les modèles pour des nombres d'itérations de la rétro-propagation du gradient égaux. Il était en effet impossible de se fier aux époques pour les comparaisons puisque le modèle possédant 5000 phrases performe la rétro-propagation 5000 fois plus que le modèle associé à l'échantillon de 1 phrase.

Pour pouvoir observer au mieux les évolutions des modèles dans les plages ou ils sont le plus amenés à changer, nous avons utilisé une fréquence d'évaluation quasi-logarithmique. Des évaluations des modèles ont été effectuées pour les étapes d'itérations suivantes : [1, 2, ..., 10, 20, ..., 100, 200, ..., 1000, 2000, ..., 10000, 20000, ... 50000]. En faisant cela, nous nous sommes assurés de capter le maximum de variations tout en minimisant le nombre d'évaluations (coûteuses) à effectuer.

La figure 4.16 montre l'apprentissage des modèles non pré-entraînés pour les différentes tailles d'entraînement et la figure 4.17 montre l'apprentissage des modèles pré-entraînés sur les treebanks de l'anglais (22000 phrases). Dans les deux cas, nous avons pris le modèle BERT multilingue.

4.4.1 Sans pré-entraînement

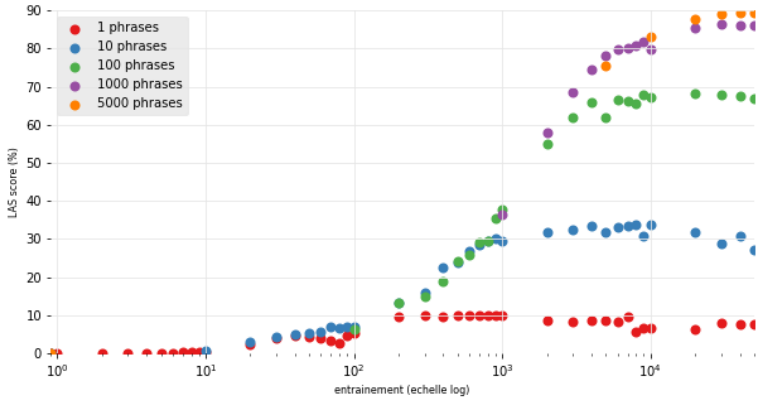

Figure 4.16: Effet de la taille du corpus d'adaptation sur l'entraînement (modèles non pré-entraînés)

Pour bien lire ce graphique, il faut bien comprendre deux choses : 1) L'échelle des abscisses est logarithmique ; 2) Un modèle s'entraînant sur n phrases n'apparaît qu'à partir de l'abscisse n. Par exemple, le modèle entraîné sur 1000 phrases (en violet), a une première époque de 1000 itérations et sa première évaluation est inscrite sur l'abscisse 10^3.

Concernant les vitesses d'apprentissage, on peut observer sur nos données un phénomène intéressant qui est que deux modèles de tailles n et $10 \times n$ ont sensiblement la même vitesse d'apprentissage sur les $10 \times n$ premières itérations (donc pour les 10 premières époques du modèle n et la première époque du modèle $10 \times n$) mais se détachent ensuite au profit du modèle $10 \times n$. Ce modèle $10 \times n$ prend ensuite le dessus est en avance au niveau de l'itération $100 \times n$.

Un autre fait intéressant est que nous pouvons voir, pour un modèle n, un plateau d'apprentissage entre $100 \times n$ et $1000 \times n$ itération (donc entre sa 100e et 1000e époque), suivi par un déclin des performances après $1000 \times n$ itérations. Le phénomène de plateau est vérifié pour les modèles $n = 1$, $n = 10$ et $n = 100$ et le phénomène de déclin pour les modèles $n = 1$ et $n = 10$. Nous ne savons pas si cela peut être généralisé puisque nous n'avons pas entraîné des modèles suffisamment grands (sur suffisamment d'époques).

Le tableau 4.9 montre les meilleures performances atteintes (POS, LUS, UAS et LAS) de ces modèles. Nous précisons que nous ne comparons pas ces résultats à ceux obtenus plus tôt dans le chapitre puisque le protocole de séparation des données est différents. Le modèle $n = 5000$ phrases fait ici "office" de modèle entraîné sur tout le jeu d'entraînement. Sans surprise, nous voyons que la taille d'entraînement d'un modèle a une influence sur sa performance maximale atteignable.

modèle	POS	LUS	UAS	LAS
1 phrases	51.32	41.18	16.14	10.07
10 phrases	82.13	72.95	39.53	33.72
100 phrases	93.38	86.17	75.28	68.15
1000 phrases	97.29	93.44	90.92	86.46
5000 phrases	97.89	94.73	93.39	89.48

Table 4.9: Performances atteintes pour les modèles entraînés sur des tailles de corpus différentes (modèles non pré-entraînés sur l'anglais)

4.4.2 Avec pré-entraînement

Pour le cas des modèles pré-entraînés (fig 4.17 et table 4.10), nous observons qu'ils ont aussi cette gradation de la performance en fonction du nombre de phrases d'entraînement. Nous observons encore le phénomène de pallier des performances entre la 100e et la 1000e époque d'entraînement des modèles et le phénomène de déclin après la 1000e époque.

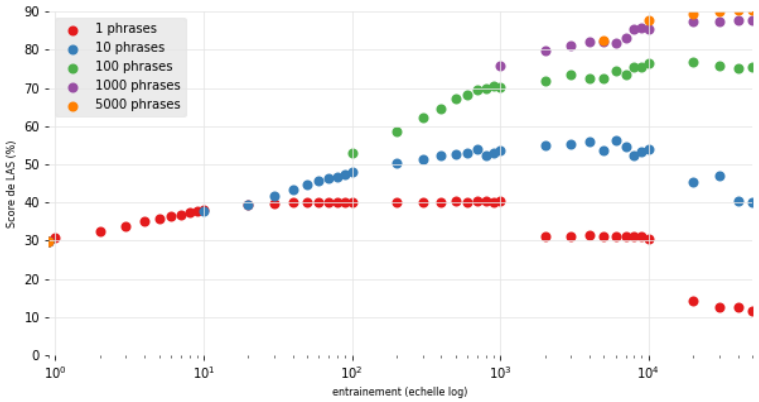

Figure 4.17: Effet de la taille du corpus d'adaptation sur l'entraînement (modèles pré-entraînés sur l'anglais)

modèle	POS	LUS	UAS	LAS
1 phrases	76.11	67.32	54.8	40.28
10 phrases	87.5	77.23	68.33	56.42
100 phrases	94.09	87.55	84.4	76.66
1000 phrases	97.48	93.23	92.55	87.76
5000 phrases	97.95	94.94	94.15	90.38

Table 4.10: Performances atteintes pour les modèles entraînés sur des tailles de corpus différentes (modèles d'abord pré-entraînés sur l'anglais)

4.4.3 Comparaison

Pour mieux se rendre compte de l'apport en performance et en vitesse d'apprentissage du pré-entraînement pour un nombre de phrases d'entraînement variable, nous avons comparé les résultats dans les figures 4.18 et 4.19 ainsi que le tableau 4.11.

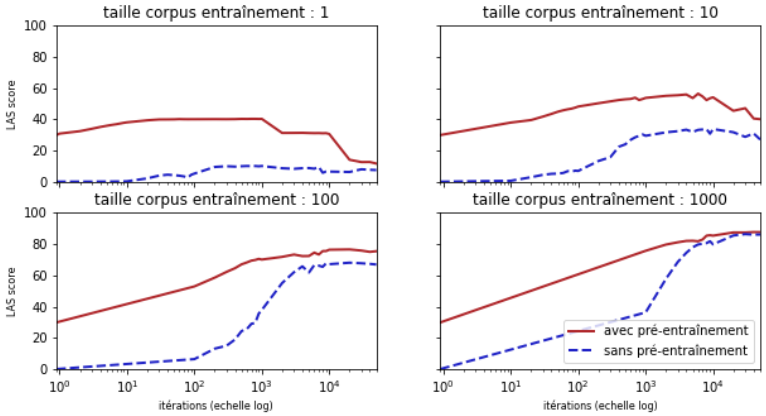

Figure 4.18: Comparaison des courbes d'entraînements entre les modèles pré-entraînés et les autres pour des tailles de corpus d'entraînements différents

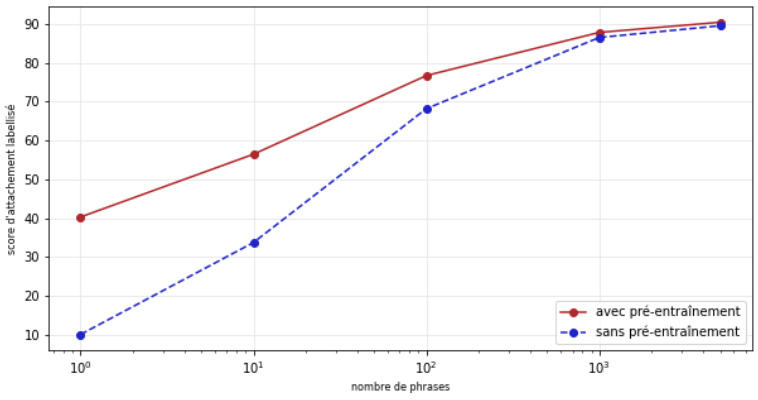

Figure 4.19: Comparaison des performances de LAS maximales atteintes entre les modèles pré-entraînés et les autres pour des tailles de corpus d'entraînements différents

modèle	LAS sans pré-entraînement	LAS avec pré-entraînement	Gain (absolu)	Gain (relatif)
1 phrases	10.07	40.28	30.21	3.0
10 phrases	33.72	56.42	22.7	0.67
100 phrases	68.15	76.66	8.51	0.12
1000 phrases	86.46	87.76	1.3	0.02
5000 phrases	89.48	90.38	0.9	0.01

Table 4.11: Table de comparaison des performances atteintes pour les modèles entraînés sur des tailles de corpus différentes

On peut voir grâce à ces figures et à ce tableau que le pré-entraînement du modèle est d'autant plus bénéfique sur les performances et la vitesse d'apprentissage que le corpus d'entraînement est petit. Le parseur pré-entraîné est 300% plus efficace que le modèle non pré-entraîné lorsque l'on dispose d'une seule phrase d'entraînement alors qu'il n'est plus que 1% plus efficace lorsque l'on dispose de plus de 1000 phrases. Ceci illustre bien l'importance décroissante de l'apprentissage par transfert supervisé sur un tâche au fur et à mesure que le corpus cible se développe.

Ces résultats sont encourageants et montrent que le processus de création de treebank peut être accéléré grandement grâce au parseur via du bootstrapping. Les performances du parseur augmentent très rapidement avec le développement du treebank si bien que 100 phrases suffisent pour atteindre une précision d'annotation de 76% (fonction + attachement syntaxique) pour les modèles pré-entraînés sur l'anglais et 68% pour les modèles bruts.

4.5 Conclusion sur l'utilité du transfert learning

Nous avons vu deux techniques qui permettent de transférer des connaissances pour notre problématique afin d'améliorer les performances et la vitesse d'apprentissage du modèle.

Premièrement, le modèle de la langue sur lequel est connecté le classifieur bi-affine permet de transférer, depuis de grands corpus bruts, de la connaissance pour notre tâche. Ce modèle de la langue, qui peut être monolingue ou multilingue, est d'autant plus efficace que ses données d'entraînement sont proches du domaine cible. Le choix de ce modèle se fait donc en fonction de la proximité linguistique entre la langue cible et les langues des modèles pré-entraînés disponibles. Dans notre cas, le BERT anglais et le BERT multilingue sont les meilleurs candidats pour notre problématique de parsing syntaxique en naija. Nous pensons que ces modèles se distinguent des autres car ils possèdent une meilleure représentation lexicale et structurelle du naija que les autres. Avec ces analyses comparatives entre ces deux modèles, nous avons mis en évidence que le modèle multilingue performe légèrement mieux que le modèle anglais grâce à un meilleur recouvrement structurel. Inversement, le modèle anglais se comporte mieux que le modèle multilingue lorsque l'on gèle les poids des modèles et cela s'expliquerait par un meilleur recouvrement lexical du modèle anglais. Le modèle multilingue posséderait un meilleur

recouvrement structurel grâce à son entraînement sur 104 langues différentes qui augmenterait les chances d'apprendre des constructions présentes du naija mais absentes en anglais. Le choix du modèle devra donc se faire au cas par cas en fonction de la langue cible, **tout en sachant que le modèle multilingue est probablement un bon candidat**.

Deuxièmement, si l'on veut préparer l'ensemble BERT + bi-affine, il est possible de pré-entraîner sur des treebanks existants le modèle afin d'initialiser ses coefficients de manière intelligente. Cela nous permet d'améliorer les performances d'adaptation du modèle et cet apport est d'autant plus significatif que les données d'adaptations sont éparses. La vitesse d'apprentissage est aussi plus rapide après un pré-entraînement supervisé et cela peut s'avérer utile lors de l'exploration des hyper-paramètres du modèle pour réduire le temps de calcul. Nous pensons que ce transfert est d'autant plus efficace que le corpus de pré-entraînement est volumineux et proche du corpus cible.

Chapitre 5

Analyse du parseur

Ce chapitre se base sur les travaux du chapitre 4 puisque nous y effectuons une analyse qualitative des prédictions du modèle choisi précédemment sur les données du treebank du naija. Nous présentons en section 5.1 une description statistique du corpus. La section 5.2 rapporte les performances macros du modèle selon plusieurs variables (longueur de la phrase, distance syntaxique, etc. . .). La section 5.3 propose une étude linguistique fine de certaines erreurs significatives et/ou intéressantes de l'analyseur.

5.1 Description préliminaire statistique

Le treebank dispose de 8259 phrases pour un total de 131042 mots (tokens). On peut voir en figure 5.1 l'histogramme de la distribution du nombre de mots par phrase. La moyenne de mots par phrase est de 15.87 (+/- 12.20 mots). Notre jeu de données comporte donc une forte densité de phrases courtes. Les phrases d'une longueur autour de 9 mots sont les plus représentées.

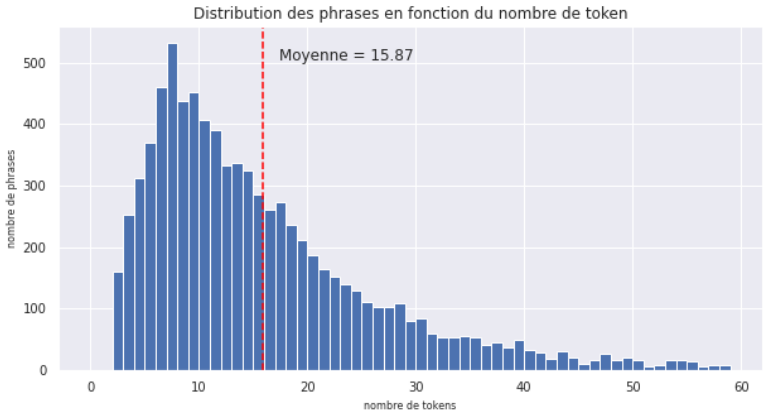

Figure 5.1: Histogramme de la distribution du nombre de mot par phrase

La figure 5.2 nous montre l'évolution de la distance syntaxique et la profondeur syntaxique

moyenne d'une phrase en fonction de la taille de la phrase. La distance syntaxique est le nombre de tokens présents entre un dépendant et son gouverneur (dépendant exclus et gouverneur inclus). La profondeur syntaxique est définie par le nombre de gouverneurs présents entre un token et la racine de la phrase. Par exemple, la racine de la phrase a une profondeur syntaxique de 0, ses dépendants ont une profondeur syntaxique de 1, et les dépendants de ses dépendants ont une profondeur syntaxique de 2 , etc.

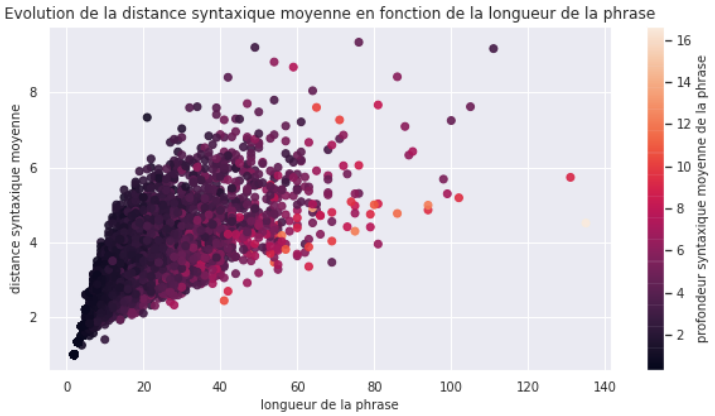

Figure 5.2: Distance syntaxique moyenne en fonction de la longueur de la phrase

Nous voyons ici que la distance syntaxique moyenne entre les mots d'une phrase est corrélée avec la longueur de la phrase. Bien que ce résultat soit intuitif, il est tout de même important de le mettre en évidence. En effet, nous aurions aussi pu observer un phénomène de chaîne ou chaque mot de la phrase est le gouverneur de son voisin de gauche (ou de droite si la chaîne est inversée) ce qui donnerait une distance syntaxique moyenne constante en fonction de la longueur de la phrase (égale à 1).

Pour nous assurer de la corrélation entre la longueur de la phrase et la distance syntaxique, nous avons calculé le coefficient de corrélation de Pearson. Nous trouvons, pour notre population de 8259 individus un coefficient de corrélation de 0.800 pour une valeur p de 0. Nous pouvons conclure que les deux variables sont **fortement corrélées positivement** sans aucun doute statistique possible.

5.2 Résultats macros de l'analyseur

5.2.1 Validation croisée

Pour pouvoir analyser les résultats de notre modèle sur la totalité du NaijaSynCor Treebank, nous avons réalisé l'entraînement de 10 modèles par validation croisée. Chaque modèle s'est entraîné sur 90% des documents et a ensuite effectué des prédictions sur les 10% restants. Pour chaque phrase du treebank, le modèle utilise l'enchaînement des tokens d'une phrase afin de prédire la catégorie du discours, le gouverneur et la fonction syntaxique de chacun de ces tokens.

Les traits morpho-syntaxiques ne sont pas pris en compte pour le calcul.

Les différents scores 5.1 sont reportés dans le tableau suivant :

	POS	LUS	UAS	LAS
Scores	96.5	94.1	94.2	90.5

Table 5.1: Scores de précisions sur l'ensemble du NSC pour les différentes mesures POS, LUS, UAS, LAS. Modèle BERT-multilingue pré-entraîné de manière non supervisé et adapté de manière supervisé sur l'anglais all$_t$*reebank*

À notre connaissance, il n'existe pas pour le moment de travaux ayant publié des scores de performances pour le parsing du naija.

5.2.2 Influence de la taille de la phrase sur la performance

Le parseur que nous utilisons fait partie de la famille des parseurs syntaxiques par graphe (McDonald, Crammer, et al. 2005). Jusqu'en 2016 avec l'invention du classifieur bi-affine (Dozat and Manning 2016), le parsing par graphe était moins performant que le parsing par transition. McDonald et Nivre ont montré en 2011 (McDonald and Nivre 2011) que les analyseurs par transition étaient plus précis sur les phrases courtes (et moins précis sur les phrases longues) que les analyseurs par graphe. Nous étudions dans cette section l'influence de la taille de la phrase sur la performance de notre modèle.

Nous avons aussi voulu étudier l'effet de la longueur de la phrase sur les performances de notre parseur du naija. Le graphique 5.3 présente, pour notre analyseur, les précisions de prédictions du gouverneur (en rouge), de la fonction syntaxique (en bleu) ainsi que de la catégorie du discours (en vert) en fonction de la longueur de la phrase. Nous avons ajouté un histogramme de la distribution des phrases en fonction de la taille de la phrase (en gris).

Puisque nous avons représenté pour chaque abscisse la valeur moyenne des erreurs des prédictions, nous ne pouvons pas nous rendre compte de la grande variance des données. Nous avons calculé et inséré dans le tableau 5.2 les coefficients de corrélation de Pearson ainsi que les valeurs p entre ces trois variables et la longueur de la phrase pour évaluer s'il y a bien une corrélation significative. Nous avons aussi inscrit les coefficients directeurs correspondant à ces couples de données (en pourcentage d'erreur par mots). Il faut bien garder à l'esprit que ces coefficients directeurs n'ont de sens que lorsque le coefficient de Pearson est non nul. Plus le coefficient de Pearson sera proche de 1 (en valeur absolue), plus le coefficient directeur apporte une information pertinente sur la direction de la courbe.

Pour la **catégorie du discours**, on voit que l'analyseur est plus mauvais pour les phrases courtes (<10 tokens) que pour les phrases moyennes (10 à 30 tokens). Le coefficient de Pearson de -0,152 nous indique qu'il y a une faible incidence négative entre la taille de la phrase et la précision de la catégorie. Nous expliquons cela par le fait que les phrases courtes ont souvent peu de contexte ce

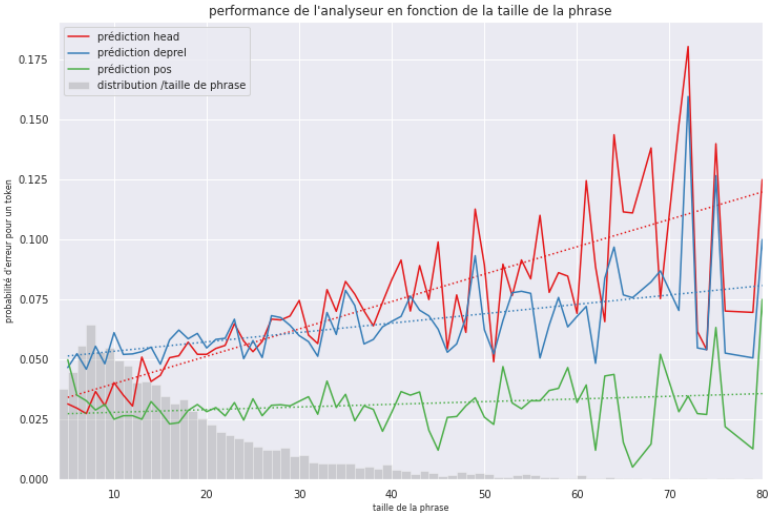

Figure 5.3: Taux d'erreur de l'analyseur en fonction de la taille de la phrase

	Coef. de Pearson	Valeur p	Coef. directeur (
prédiction du gouverneur	0.159	4.5e-48	0.0013
prédiction de la fonction	0.079	5.0e-13	0.0005
prédiction de la POS	-0.152	5.8e-44	-0.0011

Table 5.2: Taux d'erreur x (%) en fonction du nombre de mots dans la phrase

qui peut compliquer les prédictions du rôle d'un élément par rapport aux autres. Si l'on calcule à nouveau le coefficient de Pearson pour les phrases de tailles supérieures à 10, on obtient un coefficient de 0,05 (p-value = 0,003) ce qui suggère une corrélation quasi inexistante (mais non négative).

Pour la **fonction syntaxique**, la corrélation est très faible ce qui indique que la longueur de la phrase est légèrement liée au taux d'erreur de prédiction.

Pour la prédiction du **gouverneur syntaxique**, si l'on se fie à nos résultats, on a bien une augmentation du ratio d'erreur lorsque la taille de la phrase augmente. Nous avons effectué une régression linéaire sur nos 8258 échantillons et avons obtenu un coefficient directeur de 0,13 (%err/mot) par rapport à la longueur de la phrase (Cpearson=0,15, p-value=4.5e-48). Cela signifie que l'erreur de précision de la tête augmente de 1,3% pour chaque tranche de 10 mots supplémentaires. Une phrase de 60 mots aura alors un ratio d'erreur de 5% plus grand qu'une phrase de 10 mots. Pour rappel, (McDonald and Nivre 2011) avaient, à l'époque, obtenu un coefficient directeur de l'erreur par rapport à la taille de la phrase de 0,2% environ.

Les mots des phrases longues sont donc un peu plus propices à comporter des erreurs d'annotations et nous avons plusieurs hypothèses qui pourraient expliquer ce phénomène.

Une première hypothèse, informatique, est que notre corpus, comportant relativement peu de phrases longues, se spécialise sur les phrases courtes et dépendances courtes pour minimiser au mieux sa fonction de coût. De ce fait, les phrases et dépendances longues, moins présentes, ont une contribution relativement faible à l'optimisation par rétro-propagation des coefficients de l'analyseur. Pour mettre en évidence cela, nous voudrions à l'avenir effectuer un entraînement de modèle sur un corpus disposant d'une distribution uniforme par rapport à la longueur de la phrase et comparer ses résultats avec l'analyseur (plus précisément, comparer l'évolution des performances en fonction de la longueur de la phrase).

Une autre hypothèse informatique possible est qu'il pourrait y avoir un phénomène de propagation des erreurs dans l'arbre syntaxique. Lorsqu'une erreur est faite sur le gouverneur d'un nœud de l'arbre, cela pourrait propager l'erreur sur les enfants (et petits enfants) du nœud. Des arbres plus profonds auraient donc plus de chances d'avoir un début d'erreur dans une branche qui créerait alors une réaction en chaîne d'erreur chez les nœuds enfants.

Nous pensons, qu'en plus de ces deux premières hypothèses, une autre raison vient s'ajouter. Les phrases plus longues ont une grande d'avoir une distance moyenne syntaxique plus élevée (voir section 5.1), la difficulté nécessaire pour analyser les relations syntaxiques entre les tokens serait plus importante puisque le parser a désormais plus de choix pour attribuer le gouverneur de chaque token. On peut faire un parallèle avec les théories linguistiques d'augmentation de la difficulté cognitive de traitement lorsque les phrases s'agrandissent (H. Liu 2008; H. Liu et al. 2017). La figure 5.4 montre le pourcentage d'erreur en fonction de la distance au gouverneur. Nous pouvons effectivement observer que l'erreur de prédiction du gouverneur croît avec l'augmentation de la distance syntaxique entre les dépendants et leurs gouverneurs.

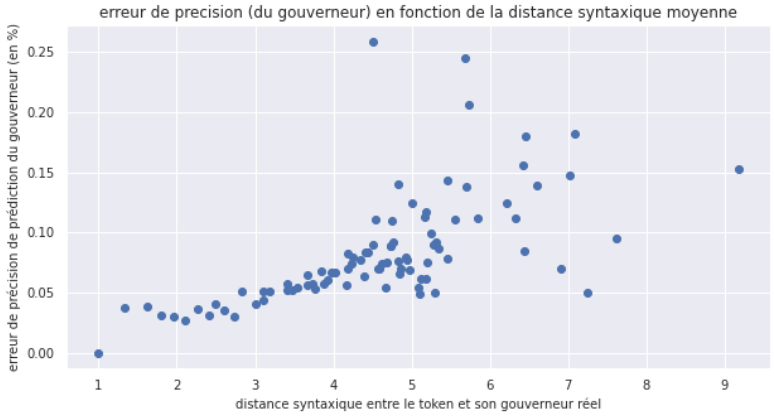

Figure 5.4: Evolution de la moyenne de l'erreur de précision du gouverneur en fonction de la distance syntaxique moyenne

Nous avons vu dans cette section que les erreurs de prédiction du gouverneur sont corrélées (faiblement) avec la taille de la phrase et que la cause pourrait être parce que les phrases plus grandes ont des distances syntaxiques plus élevées. Au vu du coefficient de Pearson qui reste relativement faible, nous devons garder à l'esprit que ce n'est pas la seule cause de variation de l'erreur.

5.3 Étude fine des erreurs d'annotations automatiques

Lors de ce projet, nous nous sommes rendu compte avec l'équipe de linguistes en charge de l'annotation que les prédictions erronées de l'analyseur apportent très généralement des informations précieuses sur l'annotation. En effet, l'analyseur apprend des régularités statistiques présentes dans le corpus d'entraînement et applique ces connaissances sur des données qui lui sont inconnues. De cette manière, on peut comparer l'annotation manuelle à la prédiction du parseur : une erreur peut alors mettre en évidence soit un problème dans l'annotation (ambiguïté du guide d'annotation ; faute ponctuelle d'inattention) soit des phénomènes linguistiques trop peu ou pas présents dans le corpus d'entraînement.

Voici certains résultats que nous avons obtenus lors de notre exploration des données.

5.3.1 Erreur en fonction de la partie du discours du dépendant

Dans un premier temps, nous nous sommes demandés quelles étaient les classes syntaxiques qui sont le moins bien appréhendées par l'analyseur pour le rattachement dépendanciel avec leurs gouverneurs. Nous avons croisé les informations concernant le succès d'attachement avec la partie du discours du dépendant ainsi qu'avec la position de ce dépendant (gauche/droite) par rapport à son gouverneur. Cette dernière information nous permet de voir si l'analyseur est aussi bon lorsqu'il est confronté à certaines exceptions du langage.

Le tableau 5.5 montre la précision d'attachement dépendanciel d'un dépendant avec son gouverneur en fonction de la partie du discours ainsi que l'agencement dépendant/gouverneur. Nous nous sommes limités à certaines parties du discours pour ne pas surcharger le diagramme et avons exclus les cas où le gouverneur est l'ancre potentielle de la phrase.

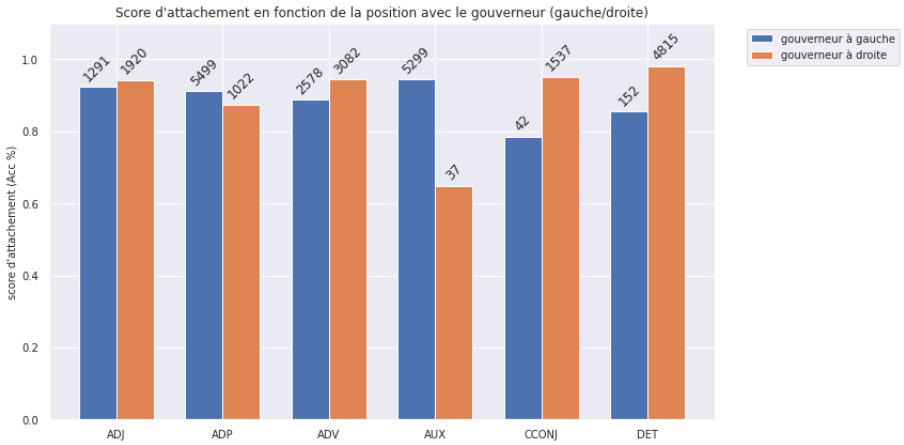

Figure 5.5: Score d'attachement syntaxique en fonction de la catégorie du discours du dépendant et de sa position avec son gouverneur. Nous avons exclu les cas ou le gouverneur est la racine *ROOT* de la phrase et n'a donc pas de POS.

Nous pouvons voir que pour chaque partie du discours représentée, l'analyseur est moins précis

pour l'agencement dépendant/gouverneur minoritaire. Cette baisse de précision est de seulement 3% pour les adjectifs et 5% pour les adpositions et adverbes, mais elle est de 10% pour les déterminants, 25% pour les conjonctions de coordination et 30% pour les auxiliaires. Pour ces derniers cas, nous pouvons observer une nette dissymétrie de la disposition dépendant/gouverneur. Cette dissymétrie pourrait causer un sur-apprentissage sur la distribution majoritaire et un sous-apprentissage sur la distribution minoritaire.

5.3.2 Erreur en fonction de la partie du discours du gouverneur

Le tableau 5.3 reporte, pour chaque partie du discours existante, le nombre d'occurrences dans le corpus, le nombre moyen de dépendants par token, le nombre d'erreurs d'attachement syntaxique parmi les dépendants et le ratio de ces erreurs parmi le nombre total de dépendants.

POS du gov	occurences	dep/gov	erreur (rel)	erreur (abs)
Moyenne	7708	1.01	17.65	422
PUNCT	37170	0.0	100.0	8
VERB	17341	2.26	4.69	1838
PRON	16827	0.17	11.06	318
NOUN	13807	1.35	7.07	1314
AUX	8635	3.34	3.46	998
ADP	6573	1.29	6.19	524
ADV	5794	0.47	8.97	244
DET	4981	0.09	15.52	72
SCONJ	4648	1.43	5.6	372
PART	4416	1.71	5.52	417
ADJ	3348	0.79	10.91	289
PROPN	2168	0.68	15.57	228
INTJ	1915	0.44	17.18	145
CCONJ	1584	0.11	38.64	68
NUM	1323	0.83	20.91	230
X	482	2.24	11.09	120

Table 5.3: Erreur en fonction de la partie du discours du gouverneur. Les colonnes représentent dans l'ordre : La POS du gouverneur, le nombre d'occurences, le nombre moyen de dépendant par gouverneur, le nombre d'erreurs en pourcentage, le nombre d'erreurs en valeur absolue

Par exemple, si l'on s'intéresse aux verbes, on peut voir qu'il y en a 17341 dans le corpus et qu'un verbe a en moyenne 2.25 dépendants (ce qui les classe deuxième derrière les auxiliaires qui ont 3.33 dépendants en moyenne). La catégorie verbe est celle qui possèdent le plus d'erreurs d'attachement syntaxique dépendanciel avec 1838 erreurs. Cependant, cela représente en valeur relative par rapport au nombre total de dépendants une valeur très faible de 4.69%. Seuls les auxiliaires ont un meilleur score de précision dépendanciel avec 3.46% d'erreurs. Les verbes (ainsi que les auxiliaires et les noms) comportent en fait relativement peu d'erreurs et ce nom-

bre absolu élevé ne dénote pas forcément d'un sous-apprentissage du parseur ou bien d'erreurs d'annotations. En revanche, la valeur d'erreurs d'attachement dépendanciel de 100% pour les ponctuations (PUNCT) a de grandes chances d'être liée à des erreurs d'annotation. En effet, les ponctuations ne peuvent être gouverneurs d'autres tokens et le parseur, en se "trompant" par rapport à l'annotation gold, nous a en fait révélé un souci d'annotation en amont que nous avons pu corriger.

5.3.3 Erreurs en fonction de la forme

Ici, nous nous intéressons aux mots les plus fréquents du corpus pour voir la régularité d'annotation du parseur. Les mots les plus fréquents d'une langue sont aussi bien souvent des mots outils (mots à usage syntaxique) qui parfois disposent de plusieurs paradigmes d'utilisation. Nous pensons que, ces mots étant largement représentés dans notre corpus, le parseur devrait pouvoir apprendre correctement du corpus et ne pas sous-apprendre (dans le cas ou il y aurait une distribution pas trop inégale des différentes utilisations de chacun de ces mots).

Le tableau 5.4 rapporte 10 des mots classés par ordre décroissant du nombre absolu d'erreurs dans le corpus (colonne 4). On peut voir en colonne 2 le nombre total d'occurrences du token dans le corpus, en colonne 6 le ratio d'erreurs d'attachement syntaxique de ce token, en colonne 3 la contribution du nombre d'erreurs d'annotation d'un token relativement à l'ensemble des erreurs du corpus (on obtient donc un pourcentage de participation aux erreurs).

	occurences	occurences	erreurs	erreurs	erreurs
	abs	ratio corpus	abs	ratio corpus	ratio token
token_form					
for	1690	1.8	167	3.39	9.88
dey	3817	4.07	150	3.04	3.93
now	705	0.75	130	2.64	18.44
go	2935	3.13	115	2.33	3.92
to	864	0.92	114	2.31	13.19
wey	1413	1.51	89	1.81	6.3
eh	403	0.43	80	1.62	19.85
na	1823	1.94	76	1.54	4.17
like	553	0.59	71	1.44	12.84
di	2630	2.8	63	1.28	2.4
so	1078	1.15	63	1.28	5.84
because	365	0.39	54	1.1	14.79
as	490	0.52	52	1.06	10.61

Table 5.4: Distribution des occurrences et erreurs des tokens en fonction de la forme

Nous pouvons observer que les cinq premiers mots de ce tableau sont responsables de pratiquement 15% des erreurs d'attachement syntaxique sur tout le treebank. Cela n'est pas à négliger

et il serait judicieux d'analyser plus en détail les erreurs sur ces mots. Il est possible que le guide d'annotation devrait davantage insister sur les mots qui entraînent le plus d'erreurs pour réduire les erreurs dans des annotations futures.

5.3.4 Confusion des fonctions syntaxiques

Les analyses faites jusqu'à présent relatent surtout des erreurs d'attachement syntaxique en fonction de différents critères comme la forme du token, sa catégorie syntaxique ou sa position dans la phrase. Nous nous sommes ici intéressés aux erreurs d'étiquetage de relations syntaxiques dans le but de révéler les cas compliqués d'analyse automatique.

En première analyse, nous avons décidé d'étudier les confusions des prédictions des fonctions syntaxiques et de présenter cela dans une matrice de confusion.

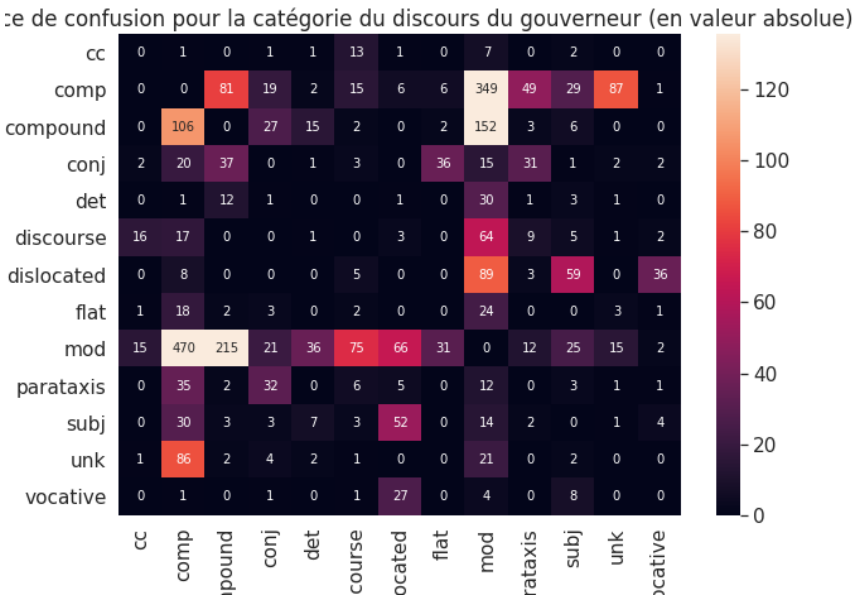

Figure 5.6: Matrice de confusion des erreurs de prédiction de gouverneur en fonction de la catégorie du discours de celui ci (en valeur absolue). Nous n'avons inclus que les cas d'erreurs et non pas les réussites.

Nous voyons sur les figures 5.6 et 5.7 des matrices avec en ligne les fonctions syntaxiques réelles et en colonne les fonctions syntaxiques prédites, la première matrice contient des valeurs absolues alors que la deuxième matrice contient des valeurs relatives (en % du total de chaque ligne). Les premières diagonales sont vides puisque nous avons seulement représenté les erreurs de l'analyseur (afin de ne pas surcharger la matrice des cas réussis qui nous sont superflus). Si nous prenons l'exemple de la fonction sujet (subj), nous pouvons voir sur la ligne correspondante de la figure 5.6 que 30 relations qui auraient dû être catégorisées *sujet* ont été catégorisées *complément* (comp), 52 ont été annotées *dislocation* (dislocated), 14 ont été annotées *modifieur*, etc...

Grâce à ces matrices, nous voyons tout de suite que les confusions critiques de l'analyseur se

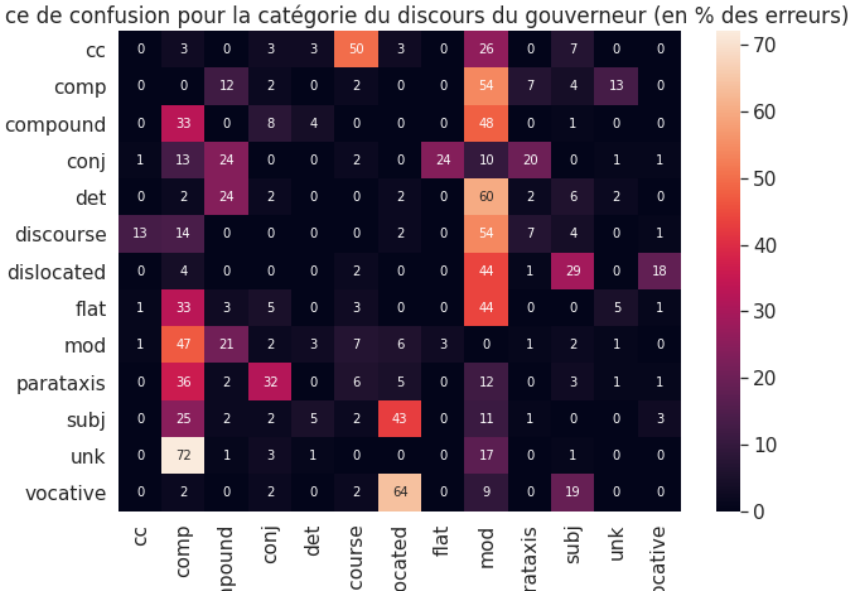

Figure 5.7: Matrice de confusion des erreurs de prédiction de gouverneur en fonction de la catégorie du discours de celui ci (en valeur relative). Nous n'avons inclus que les cas d'erreurs et non pas les réussites.

situent au niveau des *modifieurs*, des *compléments* et des *compound*. On observe plus particulièrement que, parmi les trois relations *mod comp* et *compound*, le cas de confusion relative le plus important est celui des *mod* avec *compound* que nous allons étudier dans la section suivante.

Confusion mod/compound

Suite à notre analyse des erreurs du parser, de nombreuses discussions concernant la distinction d'annotation *mod*/*compound* ont eu lieu. Ce problème d'analyse du parseur proviendrait d'une annotation non cohérente elle-même conséquence d'une convention trop ambiguë sur la définition d'une expression composée en naija. Dans le corpus annoté, il est fréquent de trouver une même composition de mots annoté de deux manières différentes. On retrouve par exemple les annotations 5.8 et 5.9 :

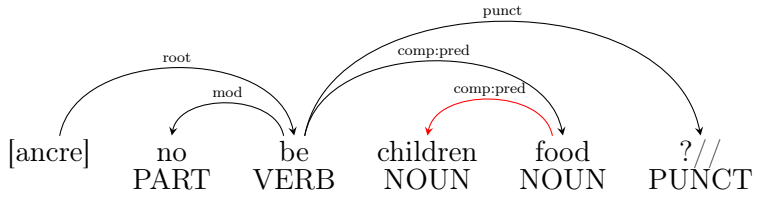

Figure 5.8: Arbre syntaxique pour la phrase en naija : *No be children food*

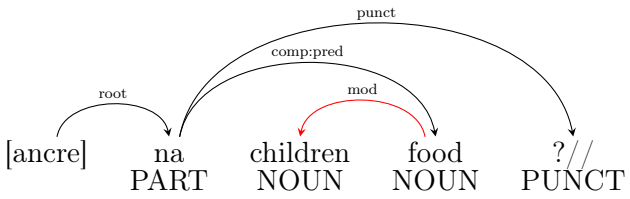

Figure 5.9: Arbre syntaxique pour la phrase en naija : *Na children food*

5.3.5 Le cas de *dey*

Nous voulons revenir sur le mot *dey* car son cas illustre bien l'importance de la cohérence de l'annotation pour obtenir un bon analyseur syntaxique. *Dey* est le mot le plus commun du naija et représente 4% de la totalité des mots (hors ponctuation) du NSCT treebank. Il possède deux rôles, celui de marqueur du présent continu (imperfectif) et celui de copule pour lier un nom avec un autre nom ou un adjectif, similaire aux formes de *to be* de l'anglais (ou de *être* du français).

Cette source de confusion se retrouve dans les résultats de la précision de prédiction du gouverneur, de la fonction syntaxique et de la catégorie du discours de *dey* que nous avons regroupées dans le tableau 5.5.

POS de dey	occurences	UAS	LUS	POS
AUX	2974	96.2	92.6	97.48
VERB	843	95.61	94.78	88.73

Table 5.5: Analyse des performances sur les prédictions associées à *dey* en fonction de ses parties du discours (AUX et VERB)

Nous observons que l'écart en précision est de 0,6% pour l'attachement syntaxique, 2,2% pour la fonction syntaxique et de 8,8% pour la catégorie du discours. De plus, la précision de la prédiction de la catégorie syntaxique lorsque *dey* est verbe (copule) est sensiblement inférieure aux deux autres scores (-6,9% / -6,1%). Cela veut dire que dans une grande partie des cas, l'analyseur prédit correctement la tête et la fonction syntaxique du mot *dey* mais échoue lors de la prédiction de la catégorie du discours (qui est normalement une tâche plus aisée que les autres.)

Nous pouvons voir ci-dessous quatre exemples d'emploi de *dey* qui ont été annotés incorrectement par l'annotateur et correctement par l'analyseur automatique.

Annotations gold=AUX /pred=VERB:

1. Deir parents *dey* rich.
2. When I *dey* higher institution. E get some people wey I be like I be *dey*

see for my school.

Annotations gold=VERB /pred=AUX:

3. You never live for ghetto before, hm mtschew you *dey* miss.
4. Dem *dey* overload and dem *dey* drive very very dangerously.

Les phrases 1) et 2) sont clairement des exemples d'utilisation de *dey* en tant que copule puisqu'il lie le sujet à l'adjectif (1) ou au groupe nominal (2). Les tokens *dey* sont donc annotés à tort en tant que AUX et auraient dû être annotés en tant que verbes (si l'on suit la convention utilisée pour le reste du corpus). Dans les phrases 3) et 4), *dey* est utilisé pour construire l'imperfectif ("you are missing out" / "they are overloading"). Encore une fois, l'analyseur révèle des erreurs d'annotations.

Nous pensons qu'il serait donc judicieux d'annoter tous les *dey* en AUX puisque cette distinction n'est pas très utile linguistiquement, qu'elle est mal prise en compte et qu'elle crée une distinction avec les conventions UD.

Lors de ce chapitre, nous avons présenté plusieurs analyses comparant les résultats de l'analyseur avec l'annotation manuelle. Grâce à la très bonne qualité du corpus initial, l'analyseur obtient de très bons résultats à la fois qualitatifs et quantitatifs. L'évaluation quantitative se fait par l'intermédiaire de plusieurs scores statistiques qui permettent de situer un modèle par rapport à d'autres. Ces scores ont l'avantage d'être objectifs et faciles à calculer ce qui permet de se concentrer sur l'ajustement du modèle pour un développement rapide. L'évolution qualitative, plus fastidieuse et moins objective, nous a montré que l'analyseur est étonnamment bon et qu'il permet, grâce à l'apprentissage statistique du corpus d'entraînement, de déceler des erreurs d'annotations. En effet, nous avons remarqué que lorsque le modèle se trompait, deux raisons sont possibles : soit il se trompe réellement ce qui indique que le modèle à sur-appris (et/ou sous-appris) et révèle certains défauts de la distribution du jeu d'entraînement ; soit il a raison et montre alors un problème d'annotation. Ces deux propriétés sont intéressantes et peuvent aider les annotateurs dans leur tâche.

Chapitre 6

Conclusion

L'objectif de notre travail était de créer un analyseur automatique de la structure syntaxique de dépendance de la phrase orale pour le naija, un pidgin-créole de l'anglais. Le corpus sur lequel est basé notre analyseur est le NaijaSynCor (projet ANR dirigé par Bernard Caron du laboratoire Llacan), un treebank annoté en syntaxe de dépendance dans le schéma SUD. Nous avons travaillé en collaboration avec les membres du projet dans le but de : 1) pré-annoter les phrases du corpus afin de préparer le travail pour les annotateurs et réduire l'effort cognitif grâce à des versions intermédiaires de l'analyseur ; 2) utiliser la version définitive de notre analyseur sur le corpus annoté manuellement pour déceler des oublis dans le guide d'annotation qui ont pu provoquer des incohérences dans l'annotation et 3) annoter automatiquement 375 000 mots supplémentaires du corpus pour former un corpus d'un total de 500 000 mots annotés en syntaxe.

Sur le plan informatique, nous avons utilisé différentes techniques récentes dans le but de créer le meilleur analyseur syntaxique du naija possible en utilisant au mieux les ressources disponibles. Nous avons utilisé un transformer (BERT) de l'anglais que nous avons adapté ("fine-tuné") au naija. Ces architectures permettent d'apprendre de manière non supervisée une certaine représentation de la langue en s'entraînant sur de très grands corpus bruts de plusieurs milliards de mots. Puisque le plus grand corpus brut du naija disponible à l'heure actuelle n'est composé que de 2 millions de mots, nous avons décidé de contourner cette étape d'entraînement d'un transformer du naija et d'utiliser un modèle déjà pré-entraîné. Nous avons donc effectué plusieurs études comparatives sur ces modèles en faisant varier certains paramètres (langue du modèle, taille du modèle, tâche d'entraînement) afin d'observer leurs influences sur le transfert de connaissances. Sans surprise, le modèle anglais est le modèle monolingue qui possède la meilleure capacité de transfert et nous pensons que cela s'explique par le fort recouvrement syntaxique et lexical des deux langues. Nous avons aussi observé que les modèles multilingues sembleraient également posséder de très bonnes capacités de transfert et seraient donc à privilégier pour toute langue n'étant pas proche d'une langue disposant d'un transformer monolingue. Il faudrait néanmoins vérifier cela pour des langues plus éloignées de l'anglais (cette dernière étant fortement présente dans le BERT multilingue pouvant donc améliorer l'utilisation du modèle sur les langues proches de l'anglais). Nous avons ensuite montré qu'il était possible d'améliorer plus profondément les

résultats en effectuant un pré-entraînement supervisé du modèle complet [BERT + analyseur] sur des treebanks de l'anglais. Bien qu'aucun de ces treebanks ne soient de l'oral ni au format SUD natif, nous observons en zero-shot un score d'attachement labellisé (LAS) de 30% de précision hors ponctuation. Nous observons donc un transfert non négligeable des connaissances apprises de l'anglais vers le naija. De plus, ces modèles pré-entraînés nécessitent un temps d'adaptation sur le treebank du naija bien inférieur aux autres. Il faut deux fois moins d'itérations pour arriver aux mêmes résultats et les paliers de score d'attachement atteints sont plus élevés de 1% en moyenne (90% -> 91%). Nous avons aussi mis en évidence une autre qualité du pré-entraînement. Plus le treebank d'entraînement du naija est petit, plus un modèle pré-entraîné sur l'anglais apporte de la connaissance relativement à un modèle non pré-entraîné. Cela peut s'avérer très utile pour les projets de création nouvelle de treebanks dès les premières annotations.

Sur le plan linguistique, nous avons contribué à l'enrichissement du plus gros treebank oral UD/SUD annoté en syntaxe toutes langues confondues ainsi qu'au plus gros treebank d'un pidgin-créole. Les incohérences entre les prédictions de l'analyseur et les annotations humaines permettent bien souvent de déceler soit des ambiguïtés dans le guide d'annotation, soit des erreurs d'annotations. Les analyses qualitatives des résultats de l'analyseur nous ont aidé à avoir un retour sur le treebank lui-même. Nous avons par exemple mis en évidence l'ambiguïté entre les fonctions syntaxiques modifieur et complément qui provient surtout d'un manque de clarté dans le guide d'annotation en amont.

L'analyseur syntaxique est maintenant utilisé pour d'autres projets linguistiques tel que la création assistée du premier Wiktionnaire du naija. Les annotations du parseur sur de grands corpus bruts permettent aux linguistes de repérer rapidement les différentes distributions syntaxiques d'un mot et d'extraire pour chaque type de construction des exemples pertinents.

La score d'attachement labellisé (LAS) atteint par notre analyseur est de 91%. Cela est très encourageant et montre entre autre la qualité du guide d'annotation SUD réalisé en amont et suivi par les annotateurs tout au long du projet. Pour approfondir nos connaissances dans ce domaine et améliorer les résultats de l'analyse syntaxique, nous avons plusieurs pistes que nous souhaitons explorer.

D'abord, nous voudrions utiliser le corpus brut du naija disponible actuellement pour améliorer les performances globales de l'analyseur. Pré-entraîner de maniere non supervisé sur ce corpus pourrait s'averer peu judicieux compte tenu de la petite taille du corpus, mais nous pourrions détourner ceci en effectuant à la place un ajustement non supervisé du modèle de la langue masqué (MLM). (Muller et al. 2020) ont montré dans leurs travaux que cette adaptation permettait d'améliorer la performance du parsing en zero-shot lors du transfert entre deux langues. Ce pré-entraînement pourrait nous permettre d'ajuster les représentations vectorielles encodées des mots présents dans le corpus bruts du naija mais relativement peu fréquents dans le treebank. Ensuite, puisque nous avons annoté un corpus de deux millions de mots provenant d'une dizaine de sources différentes (romans, écrits journalistiques, blogs, etc...), il pourrait être très intéressant, dans une perspective de 'active learning', d'identifier et ensuite de corriger manuellement certaines structures syntaxiques afin de : 1) améliorer la capacité de l'analyseur à généraliser sur de l'écrit

et 2) étudier l'influence (positive ?) sur les performances de l'analyse de l'oral après l'ajout de sources écrites.

Bibliographie

Abeillé, Anne, Lionel Clément, and François Toussenel (2003). "Building a treebank for French". In: *Treebanks*. Springer, pp. 165–187.

Ammar, Waleed et al. (2016). "Many languages, one parser". In: *Transactions of the Association for Computational Linguistics* 4, pp. 431–444.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Bakker, Peter (2008). "Pidgins versus creoles and pidgincreoles". In: *The handbook of pidgin and creole studies*, pp. 130–157.

Brown, Tom B et al. (2020). "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165*.

Carion, Nicolas et al. (2020). "End-to-End Object Detection with Transformers". In: *arXiv preprint arXiv:2005.12872*.

Caron, B (2009). "NAIJA: Proceedings of the Conference on Nigerian Pidgin". In: *University of Ibadan, Nigeria*.

Charniak, Eugene (1997). "Statistical parsing with a context-free grammar and word statistics". In: *AAAI/IAAI* 2005.598-603, p. 18.

Chen, Xinying and Kim Gerdes (2020). "Dependency Distances and Their Frequencies in Indo-European Language". In: *Journal of Quantitative Linguistics*, pp. 1–19.

Chu, Yoeng-Jin (1965). "On the shortest arborescence of a directed graph". In: *Scientia Sinica* 14, pp. 1396–1400.

Clark, Kevin et al. (2019). "What does BERT look at? an analysis of BERT's attention". In: *arXiv preprint arXiv:1906.04341*.

Courtin, Marine et al. (2018). "Establishing a Language by Annotating a Corpus". In:

Crammer, Koby and Yoram Singer (2001). "On the algorithmic implementation of multiclass kernel-based vector machines". In: *Journal of machine learning research* 2.Dec, pp. 265–292.

Devlin, Jacob et al. (2018). "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Dozat, Timothy and Christopher D Manning (2016). "Deep biaffine attention for neural dependency parsing". In: *arXiv preprint arXiv:1611.01734*.

Duong, Long et al. (2015). "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 845–850.

Dyer, Chris et al. (2015). "Transition-based dependency parsing with stack long short-term memory". In: *arXiv preprint arXiv:1505.08075*.

Edmonds, Jack (1967). "Optimum branchings". In: *Journal of Research of the national Bureau of Standards B* 71.4, pp. 233–240.

Eisner, Jason (1997). "Three new probabilistic models for dependency parsing: An exploration". In: *arXiv preprint cmp-lg/9706003*.

Firth, John R (1957). "A synopsis of linguistic theory, 1930-1955". In: *Studies in linguistic analysis*.

Futrell, Richard, Roger Levy, and Edward Gibson (2017). "Generalizing dependency distance: Comment on "Dependency distance: A new perspective on syntactic patterns in natural languages" by Haitao Liu et al." In: *Physics of life reviews* 21, pp. 197–199.

Gerdes, Kim, Bruno Guillaume, et al. (2018). "SUD or Surface-Syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD". In:

Gerdes, Kim and Sylvain Kahane (2016). "Dependency annotation choices: Assessing theoretical and practical issues of universal dependencies". In:

Guillaume, Bruno et al. (2012). "Grew: un outil de réécriture de graphes pour le TAL (Grew: a Graph Rewriting Tool for NLP)[in French]". In: *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 5: Software Demonstrations*, pp. 1–2.

Hajic, Jan, Barbora Vidová-Hladká, and Petr Pajas (2001). "The prague dependency treebank: Annotation structure and support". In: *Proceedings of the IRCS Workshop on Linguistic Databases*, pp. 105–114.

Harris, Zellig S (1951). "Methods in structural linguistics." In:

Haspelmath, Martin (2016). "The serial verb construction: Comparative concept and cross-linguistic generalizations". In: *Language and Linguistics* 17.3, pp. 291–319.

Hudson, Richard A (1984). *Word grammar*. Blackwell Oxford.

Jawahar, Ganesh, Benoît Sagot, and Djamé Seddah (2019). "What does BERT learn about the structure of language?" In:

Kahane, Sylvain (2001). "Grammaires de dépendance formelles et théorie Sens-Texte". In: *TALN 2001*.

Kahane, Sylvain and Kim Gerdes (2020). *Syntaxe théorique et formelle*.

Karthikeyan, K et al. (2019). "Cross-lingual ability of multilingual BERT: An empirical study". In: *International Conference on Learning Representations*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kiperwasser, Eliyahu and Yoav Goldberg (2016). "Simple and accurate dependency parsing using bidirectional LSTM feature representations". In: *Transactions of the Association for Computational Linguistics* 4, pp. 313–327.

Kondratyuk, Dan and Milan Straka (2019). "75 Languages, 1 Model: Parsing Universal Dependencies Universally". In: *arXiv preprint arXiv:1904.02099*.

Lacheret, Anne et al. (2014). "Rhapsodie: a prosodic-syntactic treebank for spoken french". In: *Language Resources and Evaluation Conference*.

Landauer, Thomas K and Susan T Dumais (1997). "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge." In: *Psychological review* 104.2, p. 211.

Lepikhin, Dmitry et al. (2020). "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding". In: *arXiv preprint arXiv:2006.16668*.

Liu, Haitao (2008). "Dependency distance as a metric of language comprehension difficulty". In: *Journal of Cognitive Science* 9.2, pp. 159–191.

Liu, Haitao, Chunshan Xu, and Junying Liang (2017). "Dependency distance: A new perspective on syntactic patterns in natural languages". In: *Physics of life reviews* 21, pp. 171–193.

Liu, Yinhan et al. (2019). "RoBERTa: A robustly optimized BERT pretraining approach". In: *arXiv preprint arXiv:1907.11692*.

Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz (1993). "Building a large annotated corpus of English: The Penn Treebank". In:

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

McDonald, Ryan, Koby Crammer, and Fernando Pereira (2005). "Online large-margin training of dependency parsers". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 91–98.

McDonald, Ryan and Joakim Nivre (2011). "Analyzing and integrating dependency parsers". In: *Computational Linguistics* 37.1, pp. 197–230.

McDonald, Ryan, Fernando Pereira, et al. (2005). "Non-projective dependency parsing using spanning tree algorithms". In: *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pp. 523–530.

Mel'čuk, Igor Aleksandrovic et al. (1988). *Dependency syntax: theory and practice*. SUNY press.

Mikolov, Tomas et al. (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Miyao, Yusuke et al. (2008). "Task-oriented evaluation of syntactic parsers and their representations". In: *Proceedings of ACL-08: HLT*, pp. 46–54.

Morgan, Nelson and Hervé Bourlard (1990). "Generalization and parameter estimation in feedforward nets: Some experiments". In: *Advances in neural information processing systems*, pp. 630–637.

Muller, Benjamin, Benoit Sagot, and Djamé Seddah (2020). "Can Multilingual Language Models Transfer to an Unseen Dialect? A Case Study on North African Arabizi". In: *arXiv preprint arXiv:2005.00318*.

Nivre, Joakim et al. (2016). "Universal dependencies v1: A multilingual treebank collection". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1659–1666.

O'Donovan, Ruth et al. (2005). "Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III Treebanks". In: *Computational Linguistics* 31.3, pp. 329–366.

Oh, Tae Hwan et al. (2020). "Analysis of the Penn Korean Universal Dependency Treebank (PKT-UD): Manual Revision to Build Robust Parsing Model in Korean". In: *arXiv preprint arXiv:2005.12898*.

Osborne, Timothy and Kim Gerdes (2019). "The status of function words in dependency grammar: A critique of Universal Dependencies (UD)". In: *Glossa (Online)*.

Peters, Matthew E, Sebastian Ruder, and Noah A Smith (2019). "To tune or not to tune? adapting pretrained representations to diverse tasks". In: *arXiv preprint arXiv:1903.05987*.

Reddy, Siva, Mirella Lapata, and Mark Steedman (2014). "Large-scale semantic parsing without question-answer pairs". In: *Transactions of the Association for Computational Linguistics* 2, pp. 377–392.

Ruder, Sebastian (2019). "Neural transfer learning for natural language processing". PhD thesis. NUI Galway.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Samuel, Arthur L (1959). "Some studies in machine learning using the game of checkers". In: *IBM Journal of research and development* 3.3, pp. 210–229.

Schuster, Mike and Kaisuke Nakajima (2012). "Japanese and korean voice search". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5149–5152.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015). "Neural machine translation of rare words with subword units". In: *arXiv preprint arXiv:1508.07909*.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick (2019). "BERT rediscovers the classical NLP pipeline". In: *arXiv preprint arXiv:1905.05950*.

Thompson, Neil C et al. (2020). "The Computational Limits of Deep Learning". In: *arXiv preprint arXiv:2007.05558*.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems*, pp. 5998–6008.

Weaver, Warren (1949). *Memorandum on translation*.

Yamada, Hiroyasu and Yuji Matsumoto (2003). "Statistical dependency analysis with support vector machines". In: *Proceedings of the Eighth International Conference on Parsing Technologies*, pp. 195–206.

Zhang, Yue and Joakim Nivre (2011). "Transition-based dependency parsing with rich non-local features". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 188–193.