Le modèle Transformer: un “ couteau suisse ” pour le traitement automatique des langues

François Yvon

Le modèle *Transformer*: un « couteau suisse » pour le traitement automatique des langues*

François Yvon
Université Paris-Saclay, CNRS, LISN

Février 2022

Résumé Cet article présente un survol de l'état de l'art en traitement automatique des langues, en explorant une architecture computationnelle, le modèle Transformer, qui joue un rôle central dans une large gamme d'applications. Cette architecture condense de nombreuses avancées des méthodes d'apprentissage neuronales et peut être exploitée de multiples manières : pour apprendre à représenter les entités linguistiques ; pour générer des énoncés cohérents et répondre à des questions ; pour réaliser des transformations des énoncés, une illustration étant leur traduction automatique. Ces différentes facettes de l'architecture seront successivement présentées, ce qui permettra également d'évoquer ses limitations.

Abstract This paper presents an overview of the state of the art in natural language processing, exploring one specific computational architecture, the Transformer model, which plays a central role in a wide range of applications. This architecture condenses many advances in neural learning methods and can be exploited in many ways : to learn representations for linguistic entities ; to generate coherent utterances and answer questions ; to perform utterance transformations, an illustration being their automatic translation. These different facets of the architecture will be successively presented, which will also allow us to discuss its limitations.

Mots-Clés : Traitement Automatique des Langues ; Apprentissage Automatique ; Modèles de Langues ; Traduction automatique neuronale

Keywords : Natural Language Processing ; Machine Learning ; Language Models ; Neural Machine Translation

*Cet article a fait l'objet d'une publication dans la revue "Techniques de l'Ingénieur" consultable en ligne à l'adresse https://doi.org/10.51257/a-v1-in195.

Table des matières

1 Introduction

Les technologies linguistiques figurent en bonne place parmi les applications de l'Intelligence Artificielle (IA) et touchent aujourd'hui le grand public. Elles sont essentielles pour accéder efficacement aux informations textuelles disponibles sur le Web ou dans des grandes bases documentaires ; elles permettent de nouvelles formes d'interactions avec la machine, par la voix ou par le biais de dispositifs d'aide à la saisie ou à la rédaction ; elles nous aident à communiquer avec d'autres humains, par exemple par le biais de systèmes de traduction automatique ; de manière plus souterraine, ces algorithmes structurent, organisent, filtrent, sélectionnent, transforment et rendent possible la gestion des monceaux de textes

et d'enregistrements audio qui circulent continuellement sur la toile ou sur les réseaux sociaux.

Cette transition s'est accélérée au fur et à mesure que ces technologies devenaient progressivement plus performantes pour des utilisations toujours plus larges et variées. Ces progrès résultent de la conjonction de plusieurs facteurs : d'une part le développement d'algorithmes d'apprentissage automatique de plus en plus sophistiqués, capables de tirer profit de l'amélioration des dispositifs matériel (hardware) de calcul ; d'autre part la possibilité d'accéder à de très grandes masses de données textuelles, annotées ou non annotées, pour réaliser ces apprentissages. Parmi les algorithmes, les algorithmes neuronaux et en particulier l'architecture **Transformer** figurent au premier rang. Cette architecture est en effet devenue centrale pour réaliser trois types de traitements qui jusqu'à lors nécessitaient des architectures dédiées : d'une part les algorithmes de fouille de texte et de recherche d'information, qui bénéficient de la richesse des **représentations internes** calculées par ce modèle ; ensuite les **algorithmes d'analyse linguistique** qui tirent parti de la capacité des Transformers à prendre en compte des dépendances à très longue distance ; enfin les **algorithmes de génération de texte**, qui utilisent ces modèles principalement pour leur capacité prédictive. Si l'on ajoute que cette même architecture se prête également au traitement de données orales, voire multimodale, et qu'elle permet des calculs efficaces à très grande échelle, on comprend mieux pourquoi ce modèle s'est imposé comme le véritable couteau suisse de l'ingénieur linguiste.

2 La machine à écrire : modèles de langue

2.1 Le plus simple des modèles

Considérons pour débuter une tâche élémentaire du traitement des langues : le filtrage des courriers indésirables ou pourriels. Son traitement probabiliste implique trois étapes :

1. le recueil d'un ensemble représentatifs de mails, contenant un ensemble D_{ok} de courriels acceptables et un ensemble D_{ko} de courriels indésirables ;
2. la construction d'une représentation numérique pour les textes. Une représentation très simple transforme chaque mail d en un grand vecteur binaire $\mathbf{h}$ dans $\{0,1\}^{|V|}$, avec V un vocabulaire prédéfini. Pour chaque composante, $h_w = 1$ si le mot w apparait dans le mail, 0 sinon. Ces représentations (dites « sac-de-mots ») sont intrinsèquement **creuses**, puisque la plupart des composantes de ce vecteur sont nulles ;
3. l'apprentissage d'un modèle probabiliste [1] $P(\text{OK}|d) \propto \exp \sum_w \theta_w h_w$, qui évalue la probabilité qu'un courriel soit désirable. Le vecteur de para-

1. La notation $P(u|x) \propto \exp f(u, x, \boldsymbol{\theta})$ indique que la probabilité conditionnelle d'observer la réalisation u quand on observe x est *proportionnelle à* $\exp f(u, x, \boldsymbol{\theta})$, une grandeur positive parfois appelé *logit*. Pour obtenir $P(u|x)$ il faut normaliser ce terme en le divisant par la somme sur toutes les réalisations possibles $\sum_{u'} f(u', x, \boldsymbol{\theta})$.

mètres $\boldsymbol{\theta}$ permet de pondérer la contribution de chacun des mots à la décision finale. La présence d'un mot pour lequel θ_w est positif renforce la probabilité d'un mail correct ; inversement les mots pour lesquels θ_w est très négatif l'atténuent et inversement renforcent la probabilité d'un pourriel. L'apprentissage de $\boldsymbol{\theta}$ est réalisé en maximisant la log-vraisemblance des données d'apprentissage, selon :

$$\ell(\boldsymbol{\theta}) = \sum_{d \in D_{\text{ok}}} \log P(\text{OK}|d) + \sum_{d \in D_{\text{ko}}} \log(1 - P(\text{OK}|d)).$$

Cette application historique des technologies linguistiques se décline sous de multiples formes : routage et classification multiclasse de documents, analyse de « sentiment » ou d'opinion (les classes correspondent à la polarité du texte), implication textuelle, visant à décider si une phrase implique logiquement une autre phrase, etc. Elle permet de mettre en évidence trois concepts essentiels de la démarche qui s'est imposée depuis les années 1990 pour aborder les problèmes de TAL : (a) le calcul de **représentations numériques** (ici binaires) des entités linguistiques et de leurs propriétés ; (b) l'utilisation de ces représentations dans des modèles probabilistes évaluant des **décisions discrètes** (ici : classer un mail dans une parmi deux classes possibles) ; l'apprentissage des paramètres de ces modèles en exploitant des **données annotées** (ici : des courriels corrects et incorrects). Comme nous le verrons, les évolutions les plus récentes du domaine continuent de reposer sur ces concepts, utilisant les réseaux de neurones pour apprendre des représentations et des modèles incomparablement plus sophistiqués que celui esquissé ci-dessus.

2.2 De l'ordre des mots

Pour le filtrage des mails, il est possible de faire abstraction de l'ordre des mots et, plus globalement, de la structure des énoncés linguistiques, pour construire des représentations utiles. Ces représentations ignorent pourtant une des propriétés essentielles des textes, à savoir leur organisation en une séquence linéaire d'unités [2]. Les **modèles de langue** sont des modèles probabilistes conçus pour prendre en compte cette séquentialité.

Nous notons $\mathbf{w} = w_1 \dots w_T$ une séquence discrète comprenant T unités (mots) notées w_t. Dans un modèle de langue n-gramme, la probabilité de cette suite s'écrit :

$$\begin{array}{rcl} P(w_1 \dots w_T) & = & \prod_{t=1}^{T} P(w_t|w_1 \dots w_{t-1}) \\ & = & \prod_{t=1}^{T} P(w_t|w_{t-n+1} \dots w_{t-1}). \end{array} \quad (1)$$

La première ligne décompose la probabilité de la séquence comme un produit de distributions conditionnelles ; la seconde rend cette décomposition tractable en faisant une **hypothèse de localité des dépendances**, qui stipule que

2. En fait une **triple** séquence d'unités : des suites de sons ou de lettres composant des mots, des suites de mots composant des énoncés, des suites d'énoncés composant un discours ou un dialogue.

la probabilité d'apparition de l'unité w_t est indépendante du passé sachant le **contexte** constitué des $n-1$ mots précédents. Les distributions conditionnelles correspondantes sont des probabilités discrètes qui constituent ici les paramètres $\boldsymbol{\theta}$ du modèle : en supposant que de surcroit le vocabulaire V est fini et connu à l'avance, ces paramètres sont en nombre fini. Conceptuellement, ce modèle est identique au modèle précédent : il permet d'assigner un « document » (ici réduit au quelques mots qui précèdent la position courante) à une « classe » (ici, un mot parmi tous les mots possibles). Le modèle n-gramme admet des procédures d'estimation efficaces qui s'appuient sur des décomptes d'occurences dans des grands corpus et prennent la forme suivante pour un modèle d'ordre 2 (aussi appelé **modèle trigramme**) :

$$\forall u, v, w : P(w|uv) = \frac{n(uvw)}{\sum_{w' \in V} n(uvw')}, \tag{2}$$

où $n(uvw)$ dénombre le nombre d'occurrences de la séquence uvw dans un corpus d'apprentissage.

Les deux hypothèses de base des modèles n-grammes (dépendances locales, vocabulaire fini) sont linguistiquement simplistes. Il existe d'une part de multiples exemples d'influence entre mots distants, que cette influence soit d'ordre syntaxique (*les décisions de ma cheffe de service sont efficaces*, où l'accord pluriel se fait entre *décisions* et *sont*, séparés pourtant par quatre mots), sémantique (*les juges de la cour européenne de justice ont décidé*, où *décider* peut être prédit en tant qu'action typique réalisée par des *juges*), ou encore d'ordre discursif, thématique ou stylistique. Il existe, d'autre part, de multiples arguments qui s'opposent à l'idée de finitude du vocabulaire : nous revenons sur cette question à la section 2.4.

Malgré leur simplicité, les modèles de langue sont utiles pour une grande gamme d'applications. En premier lieu, il est possible de s'en servir comme des **générateurs automatiques** de textes : il suffit pour cela d'utiliser l'équation (1) de manière répétitive, en tirant à chaque étape le prochain mot, conditionnellement aux mots précédemment sélectionnés. En second lieu, ces modèles permettent de comparer entre elles plusieurs séquences afin de sélectionner celle qui est la plus plausible : ce sera souvent celle qui est la plus correcte grammaticalement. La capacité à prendre ces décisions sera par exemple utile à un correcteur d'orthographe, qui doit choisir la meilleure correction ; ou encore à un traducteur, pour sélectionner l'hypothèse de traduction qui semble la plus correcte, etc. En troisième lieu, ils sont utiles pour **comparer des langues** : si l'on dispose d'un modèle de langue appris pour des textes français et d'un autre appris sur des textes italiens, comparer les probabilités d'une phrase pour ces deuxs modèle fournit un moyen de décider quelle est la langue du texte la plus probable. Ils permettent également de modéliser d'autres types de séquences linguistiques : des séquences de sons, de lettres, ou bien encore des séquences d'arguments pour modéliser des discours.

Popularisés par leur efficacité en traitement de la parole [Jelinek and Mercer, 1980, Jelinek, 1997], les modèles de langue sont très tôt devenus des outils essentiels pour le traitement statistique des langues et ont donné lieu à d'innombrables développements, portant notamment sur l'amélioration des procédures d'estimation. Les estimateurs obtenus par décompte (équation (2)) ne sont en fait pas appropriés pour rendre compte de la probabilité d'évènements très rares. Dès que l'on considère des grands vocabulaires de plusieurs dizaines de milliers d'unités, la grande majorité des suites de deux ou trois mots ne sont jamais observées, et utiliser les rapports de fréquences conduit à estimer des valeurs nulles pour la majorité des paramètres. Les méthodes de **lissage** (*smoothing*) visent alors à améliorer ces estimateurs en s'appuyant par exemple sur des regroupements de mots. Un bilan de ces développements est dressé par [Rosenfeld, 2000] ; des généralisations de ce modèle s'appuyant sur les modèles de Markov ou sur les grammaires stochastiques sont détaillées dans [Charniak, 1993] ; des introductions récentes à ces techniques se trouvent dans des ouvrages de référence sur le TAL [Manning and Schütze, 1999, Jurafsky and Martin, 2000, Eisenstein, 2019].

2.3 Modélisations neuronales : complexification du contexte

2.3.1 Un réseau multi-couche pour les modèles de langue

Prédire le prochain mot à partir des quelques mots qui précèdent est formellement équivalent à un problème de classification (de documents), puisqu'il s'agit de prédire un symbole (dans l'ensemble V) à partir du « document » constitué des mots précédents. Les auteurs de [Bengio et al., 2003] proposent de le traiter avec un réseau à propagation directe (*feedforward network*) reproduit à la Figure 1 pour un modèle d'ordre 3.

L'entrée du réseau calculant $P(w|tuv)$ comprend les trois vecteurs $\mathbf{t}, \mathbf{u}, \mathbf{v}$ dans $\{0,1\}^{|V|}$ où les mots t, u, v sont remplacés par des vecteurs binaires dont la seule composante non nulle est l'indice du mot dans le vocabulaire (on parle d'encodage *one-hot*). Le calcul se déroule ensuite selon [3] :

$$\begin{aligned}
\mathbf{i}^T &= [\mathbf{t}^T\mathbf{R}; \mathbf{u}^T\mathbf{R}; \mathbf{v}^T\mathbf{R}], \text{ avec } \mathbf{R} \in \mathbb{R}^{|V|\times d_{\text{md}}} \\
\mathbf{h}^T &= \phi(\mathbf{i}^T\mathbf{W_{ih}} + \mathbf{b}_{ih}^T), \text{ avec } \mathbf{W}_{ih} \in \mathbb{R}^{3d_{\text{md}}\times d_{\text{md}}} \text{ et } \mathbf{b}_{ih} \in \mathbb{R}^{d_{\text{md}}} \\
\mathbf{o}^T &= \mathbf{h}^T\mathbf{W}_{ho} + \mathbf{b}_{ho}^T, \text{ avec } \mathbf{W_{ho}} \in \mathbb{R}^{d_{\text{md}}\times|V|} \text{ et } \mathbf{b}_{ho}^T \in \mathbb{R}^{|V|} \\
P(w|tuv) &= \text{softmax}(\mathbf{o})_w, \text{ avec softmax}(\mathbf{x})_t = \frac{\exp(x_t)}{\sum_{t'}\exp(x_{t'})}
\end{aligned} \tag{3}$$

Ces quatre étapes correspondent respectivement :

1. au calcul de **représentations numériques denses**, via la matrice $\mathbf{R}$

3. Pour un vecteur colonne $\mathbf{x}$ de dimensions $(d \times 1)$, $\mathbf{x}^T$ désigne le vecteur ligne transposé de dimension $(1 \times d)$

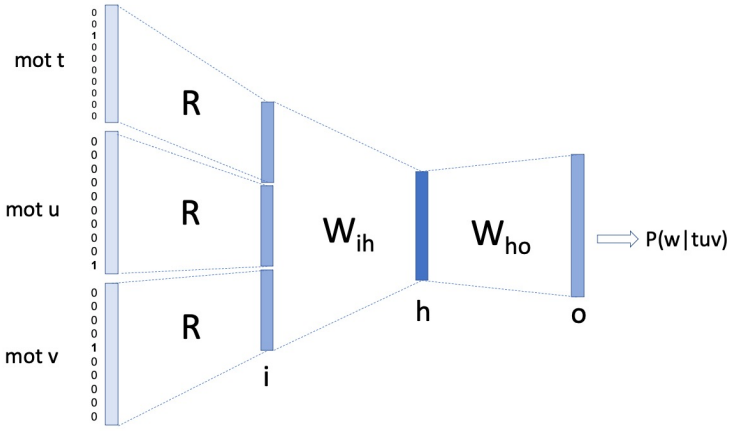

FIGURE 1 – Réseau multi-couche implantant un modèle 4-gramme

qui projette chaque vecteur d'entrée dans un espace de dimension d_{md}, avec $d_{\text{md}} \ll |V|$;

2. à l'introduction d'une ''non-linéarité", via la fonction $\phi()$, réalisée par la fonction tangente hyperbolique (tanh) dans l'implantation originale ;
3. au calcul de scores (*logits*) non-normalisés pour chacun des mots pouvant suivre le contexte tuv, obtenus en comparant la sortie de la couche cachée avec les représentations lexicales de sortie $(\mathbf{W}_{ho})$ [4] ;
4. à la normalisation de ces scores via l'opérateur softmax, qui produit un vecteur de probabilités.

L'apprentissage de ces modèles demande d'utiliser des méthodes d'optimisation numérique qui ajustent les paramètres $\boldsymbol{\theta} = \{\mathbf{R}, \mathbf{W_{ih}}, \mathbf{b_{ih}}, \mathbf{W_{ho}}, \mathbf{b_{ho}}\}$ afin de rendre probables les associations entre contextes et mots observées sur un grand corpus. Formellement, pour chaque séquence $[t, u, v, w]$ du corpus d'apprentissage, on souhaite que la quantité $\log P(w|tuv) = o_w - \log \sum_{w'} \exp o_{w'}$ soit la plus grande possible. Ceci conduit à maximiser (en $\boldsymbol{\theta}$) le critère suivant **d'entropie croisée** :

$$\ell(\boldsymbol{\theta}) = \sum_{[t,u,v,w]} \log P(w|tuv) \tag{4}$$

$$\text{avec } \forall u, t, u, v, \log P(w|tuv) = o_w - \log \sum_{w'} \exp o_{w'} \tag{5}$$

Cette optimisation est usuellement réalisée par des méthodes numériques de

4. Il est tout à fait possible d'utiliser dans la sortie une représentation lexicale différente de celle utilisée pour l'entrée.

descente de gradient stochastique, qui mettent à jour les paramètres à partir des valeurs du gradient. Notons que cet apprentissage ne requiert aucune annotation du corpus et peut être réalisé sur d'immenses quantités de textes, dès lors que l'on sait les segmenter en ''mots". Pour le modèle trigramme, il suffira ainsi d'extraire toutes les suites de quatre mots ainsi que leur fréquence.

Le passage de modèles discrets à des représentations numériques est computationnellement intensif, car le calcul de l'opérateur softmax implique une somme sur un grand vocabulaire. Des solutions pratiques sont proposées et évaluées dans [Schwenk, 2007, Le et al., 2011], nous en rediscuterons à la section 2.4. Ce passage s'est toutefois avéré décisif pour améliorer la qualité d'applications comme la reconnaissance vocale ou la traduction automatique. Il permet d'apprendre simultanément :

(a) : une fonction qui calcule une représentation numérique $\mathbf{h}$ résumant le contexte (les mots précédents) dans un vecteur de faible dimension, à partir duquel la distribution conditionnelle des mots successeurs est calculée. Cette capacité à calculer des représentations numériques du contexte de prédiction est assimilée à la **fonction d'encodage** du réseau neuronal. (b) : une projection (un **plongement lexical**) du vocabulaire V dans $\mathbb{R}^{d_{\text{md}}}$ à travers la matrice $\mathbf{R}$. Ce plongement possède des propriétés remarquables ; il permet en particulier de rapprocher les représentations de mots partageant des contextes communs, qui sont souvent des mots sémantiquement apparentés ou des mots d'une même famille morphologique. L'utilisation de ces plongements comme représentations lexicales génériques [Collobert et al., 2011] s'est généralisée avec le développement de méthodes rapides et efficaces pour les calculer [Mikolov et al., 2013, Bojanowski et al., 2016], prenant une part de plus en plus importante dans les développements du domaine.

2.3.2 Récursion linéaire

Le modèle précédent partage avec le modèle n-gramme l'utilisation d'un contexte restreint aux $n-1$ mots voisins. Le recours à des **réseaux récurrents** [Elman, 1990], permet de passer outre cette limitation et donc de calculer des termes $P(w_{t+1}|w_1 \dots w_t)$ **sans faire l'hypothèse de dépendances locales**. L'intérêt de cette approche et sa supériorité sur le modèle précédent sont mis en évidence dans [Mikolov et al., 2010], qui présente un réseau capable de prendre en compte des contextes arbitrairement longs. On y retrouve les deux mêmes composants que précédemment, à savoir : (a) une représentation numérique dense des unités lexicales calculée par la matrice $\mathbf{R}$; (b) une fonction d'encodage du contexte, calculée ici récursivement par $\phi()$, qui de nouveau dénote une fonction non-linéaire :

$$\begin{aligned} \mathbf{h_t} &= \phi(\mathbf{W_{h'h}h_{t-1}} + \mathbf{Rw_t} + \mathbf{b_h}) \\ &= \phi(\mathbf{W_{h'h}}\phi(\mathbf{W_{h'h}h_{t-2}} + \mathbf{Rw_{t-1}} + \mathbf{b}_h) + \mathbf{Rw_t} + \mathbf{b}_h). \end{aligned} \tag{6}$$

Comme précédemment, il reste à projeter la représentation interne $\mathbf{h}_t$ pour

calculer la distribution conditionnelle de sortie associée au contexte $w_{\leq t} = w_1 \dots w_t$ selon $P(w|w_{\leq t}) = P(w|\mathbf{h_t}) = \text{softmax}(\mathbf{W}_{ho}\mathbf{h_t} + \mathbf{b_o})$. On entraine les paramètres du réseau récurrent $\{\boldsymbol{\theta} = \mathbf{R}, \mathbf{W_{h'h}}, \mathbf{W_{ho}}, \mathbf{b_h}, \mathbf{b_o}\}$ en maximisant la somme sur le corpus des log-probabilités.

Déplier la récursion (seconde ligne de l'équation (6)) explicite la relation fonctionnelle entre $\mathbf{h}_t$ et les mots $\mathbf{w}_t$ et $\mathbf{w}_{t-1}$, puis par récurrence, avec tous les mots précédents. Ceci révèle également que l'influence des mots diminue au fur et à mesure qu'ils sont plus éloignés de la position courante. Une difficulté d'ordre computationnel est associée à cette observation. Le calcul direct du gradient (par les règles de dérivation de fonctions composées) des expressions précédentes donne lieu à des instabilités numériques qui rendent l'apprentissage plus délicat. Des remèdes, qui impliquent l'utilisation de dépendances plus complexes entre $\mathbf{h_t}$ et $\mathbf{h_{t-1}}$ sont proposés par [Hochreiter and Schmidhuber, 1997, Cho et al., 2014b]. Ils permettent d'exprimer le plein potentiel de ces réseaux, qui parviennent à capturer certaines dépendances entre mots distants – comme celle que l'on observe en français entre le sujet et son verbe, qui doivent s'accorder en nombre et en personne indépendamment de leur éloignement dans la phrase (voir [Linzen et al., 2016] pour une étude de ces phénomènes). Leur expressivité en tant que modèle de calcul est analysée en détail dans [Merrill et al., 2020].

À l'usage, la formulation récursive du calcul des représentations latentes pose toutefois un problème majeur : le traitement d'une séquence, que ce soit à l'apprentissage ou à l'inférence, demande qu'elle soit traitée mot après mot dans l'ordre de leur apparition. Il est impossible de construire $\mathbf{h_t}$ sans avoir préalablement calculé $\mathbf{h_{t-1}}$ qui lui même requiert $\mathbf{h_{t-2}}$ etc. ; on parle alors de **modèle autorégressif**. En conséquence, il n'est pas possible de traiter en parallèle les mots d'une phrase pour calculer la fonction objectif (équation (4)), ce qui ralentit considérablement l'entrainement sur de très gros corpus.

2.3.3 Les réseaux récurrents comme représentations : ELMo

Parmi les nombreuses extensions de ces modèles, la plus remarquable est leur utilisation en tant que "purs" encodeurs. Observons tout d'abord que l'apprentissage de ces modèles de langue particuliers ne demande aucune annotation et peut donc être réalisé sur des très gros corpus de données disponibles sur la Toile. Supposant les paramètres connus, un réseau récurrent transforme une suite de mots $w_1 \dots w_T$ en une suite de vecteurs $\mathbf{h}_1 \dots \mathbf{h}_T$. Le même processus peut être réalisé en parcourant les séquences à l'envers, depuis w_T jusqu'à w_1, ce qui permet d'obtenir $\tilde{\mathbf{h}}_\mathbf{1} \dots \tilde{\mathbf{h}}_\mathbf{T}$. Concaténer les deux représentations courantes produit le vecteur $[\mathbf{h}_t; \tilde{\mathbf{h}}_t]$, qui représente le mot w_t dans un *contexte bidirectionnel*, intégrant à la fois les mots précédents et suivants. Il s'avère également que $[\tilde{\mathbf{h}}_\mathbf{1}; \mathbf{h_T}]$ est une très bonne manière de représenter dans un vecteur de taille fixe l'ensemble de la phrase $w_1 \dots w_T$, qui est pourtant de taille variable). Ce vecteur peut alors être utilisé pour comparer des phrases ou encore pour faire des prédictions relatives à leur polarité ou leur sens. Il s'avère également qu'en utilisant la concaténation $[\mathbf{h}_t; \tilde{\mathbf{h}}_t]$ comme entrée d'une nouvelle couche de calcul,

les représentations (profondes) apprises sont encore de meilleure qualité. Ces principes seront utilisés pour construire le modèle ELMo [Peters et al., 2018], qui est l'un des premiers à mettre en évidence la richesse de ces représentations contextuelles profondes et à proposer de les utiliser comme un pré-traitement applicable et bénéfique pour toute application traitant des séquences d'unités linguistiques.

En guise d'illustration, considérons la tâche **d'implication textuelle**, qui consiste à décider si une phrase $\mathbf{w}_P$ implique logiquement une seconde phrase $\mathbf{w}_C$. Pour modéliser cette tâche, il faut prédire une réponse binaire (Oui ou Non) à partir de deux phrases, donc modéliser $P(\text{Oui} \,|\mathbf{w}_P, \mathbf{w}_C)$. Une approche possible encodera chaque phrase en un vecteur unique (respectivement $\text{ELMo}(\mathbf{w}_P)$ et $\text{ELMo}(\mathbf{w}_C)$) qui sont concaténés et utilisés dans un modèle log-linéaire par exemple selon :

$$P(\text{Oui} \,|\mathbf{w}_P, \mathbf{w}_C) \propto \exp(\mathbf{W}[\text{ELMo}(\mathbf{w}_P); \text{ELMo}(\mathbf{w}_C)] + \mathbf{b}),$$

où la matrice $\mathbf{W}$ et le vecteur $\mathbf{b}$ constituent les paramètres du modèle d'implication. En pré-entrainant les paramètres du modèle ELMo, puis en entrainant ceux du modèle d'implication textuelle, il est possible d'obtenir de très bonnes performances même lorsque les données d'entrainement du modèle d'implication textuelle sont limitées.

2.4 Questions de vocabulaire

Nous avons laissé en suspens la question du support des distributions de probabilité représentées par les équations (1) et (3). Elles présupposent l'existence d'un inventaire fini V d'unités discrètes. Pour modéliser des séquences de lettres, de sons ou de syllabes, cette hypothèse est facile à défendre. Pour des séquences de mots, elle n'est plus tenable, puisqu'aucun corpus, aussi large soit-il, ne pourra épuiser les mécanismes de création lexicale (réguliers ou irréguliers), sans parler des emprunts à d'autres langues, et des extra-lexicaux (noms propres, chiffres, sigles) dont il faut modéliser les occurrences. Cette question a longtemps été source de difficulté pour les modèles de langue et a justifié la prise en compte de vocabulaires de très grande taille, en dépit des problèmes computationnels associés.

Computationnellement, un meilleur compromis est réalisé en abandonnant la notion de mot et en segmentant les textes en **unités sous-lexicales**, par des procédés qui sont eux-mêmes optimisés sur corpus pour prendre en compte les fréquences d'occurrence. Les mots fréquents sont ainsi préservés dans leur intégrité, tandis que les mots les plus rares sont divisés en sous-mots, le cas échéant réduits à des suites de symboles élémentaires. À ce prix, il est possible de manipuler des vocabulaires d'unités de taille moyenne (quelques dizaines de milliers d'unités), tout en se donnant la possibilité de probabiliser des séquences de mots arbitraires (comprenant des mots inconnus, construits par agrégation de sous-mots connus). Les algorithmes les plus connus pour réaliser ce traitement

sont l'algorithme *Byte Pair Encoding* (BPE) [Gage, 1994, Sennrich et al., 2016] et l'algorithme *unigram* [Deligne and Bimbot, 1995, Kudo, 2018]. Des exemples de segmentations réalisées par ces algorithmes sont reproduits à la Figure 2.

```
_tous _les _êtres _humains _n aissent _libres _et _ég aux _en _dign ité _et _en
_droits . _Ils _sont _dou és _de _raison _et _de _conscience _et _doivent _agir
_les _uns _envers _les _autres _dans _un _esprit _de _fra tern ité .

_all _human _b e ings _a re _bor n _fre e _and _e qu al _in _dign ity _and _ri
gh ts .

_alle _M ens ch en _sin d _fre i _un d _g le ich _an _W ü r de _un d _Re ch
ten _g eb or en .
```

FIGURE 2 – Segmentation en unités sous-lexicale du premier article des versions respectivement française, anglaise et allemande de la Déclaration Universelle des droits humains. Le vocabulaire contient 10000 unités, le caractère '_' identifie les unités en début de mots. Avec ce modèle de segmentation optimisé sur des textes français, seuls les mots rares (par exemple : 'dignité', 'fraternité') sont segmentés. Il peut également servir à segmenter **avec le même ensemble d'unités** des textes rédigés dans d'autres langues utilisant le même alphabet, comme ici des phrases en anglais et en allemand.

2.5 À retenir :

- Les modèles de langue permettent de calculer la probabilité d'une séquence ordonnée d'unités discrètes ;
- Leur apprentissage ne demande aucune annotation et s'appuie sur la tâche de prédiction du mot suivant ;
- Ils sont utilisés pour engendrer des séquences, ou encore pour évaluer la vraisemblance linguistique d'une séquence produite par un autre algorithme ;
- Les modèles de langue neuronaux intègrent une fonction supplémentaire : le calcul de représentations numériques denses pour les unités de la langue ;
- Les modèles récurrents permettent de plus de prendre en compte des relations d'influence à longue distance entre mots ;

3 Le modèle Transformer

3.1 L'attention, un mécanisme fondamental

Ces premiers concepts de la modélisation probabiliste étant établis, nous introduisons dans cette section le modèle Transformer, qui repose sur un mécanisme plus général pour encoder le contexte de chaque décision.

3.1.1 Calculer le vecteur contexte

L'idée centrale du modèle Transformer [Vaswani et al., 2017] consiste à faire dépendre la représentation du mot w_t de tous les mots de son contexte gauche selon $\mathbf{h}_t = \phi(\mathbf{w}_1 \dots \mathbf{w}_t)$, tout en faisant disparaître la récurrence du calcul de $\phi()$ afin de pouvoir le paralléliser. Dans le modèle Transformer, ce calcul est réalisé par un empilement de L couches de calcul. Chaque couche l recombine les représentations issues de la couche précédente $\mathbf{h}_1^{(l-1)} \dots \mathbf{h}_t^{(l-1)}$ pour constuire des sorties $\mathbf{h}_1^{(l)} \dots \mathbf{h}_t^{(l)}$ en exploitant des opérations élémentaires : la projection linéaire, la combinaison linéaire, la concaténation de vecteurs, plus des réseaux à propagation avant. La récursion du modèle récurrent (équation (6)), qui réalise un empilement temporel des états cachés par lequel l'influence des mots plus lointain est plus diffuse, est ainsi remplacée par un empilement de couches de calcul ayant chacune une portée globale. Le résultat reste le même que pour les autres modèles de langue : une représentation vectorielle du contexte qui résume l'ensemble des mots précédents, à partir de laquelle on prédit le prochain mot de la séquence. La figure 3 illustre les encodages du contexte réalisés par ces différentes architectures neuronales.

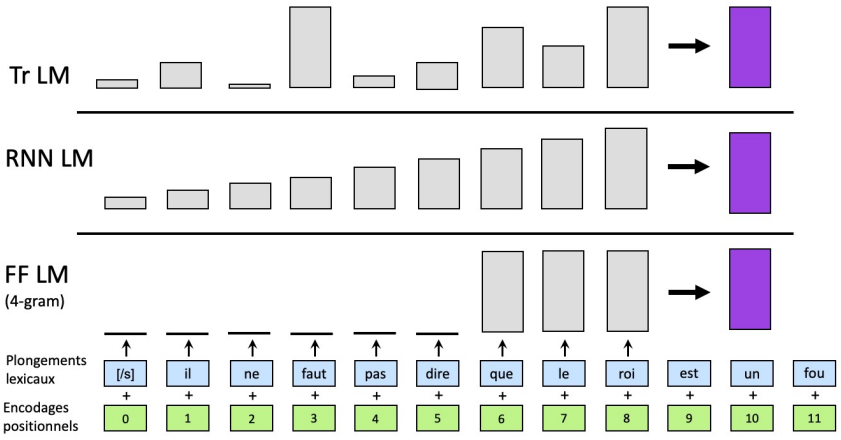

FIGURE 3 – Les encodage du contexte gauche réalisés par divers modèles de langue : les modèles n-gramme (FF LM) n'encodent qu'un contexte réduit ; les modèles RNN accordent plus d'importance aux mots les plus proches ; les Transformers (Tr LM) apprennent un encodage qui traite à égalité tous les mots du contexte.

Formellement, chaque couche d'un Transformer est paramétrée par un ensemble de K **têtes d'attention**, auxquelles s'ajoute un perceptron multi-couche. Le comportement de chaque tête d'attention est déterminé par trois matrices de projection $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ dans $\mathbb{R}^{d_{\text{md}} \times d_{kv}}$ et réalise les deux calculs suivants pour calculer

$\mathbf{h}_t^{(l)}$ à partir des sorties de la couche précédente $\{\mathbf{h}_s^{(l-1)}, s = 1 \dots t\}$. Pour la $k^{\text{ème}}$ tête de la $l^{\text{ème}}$ couche :

$$
\begin{array}{llrcl}
(7.q) & \text{Requête} & \mathbf{q_t}^{(k,l)} & = & \mathbf{Q}^{(k,l)}\mathbf{h_t}^{(l-1)} \quad (\in \mathbb{R}^{d_{\text{kv}}}) \\
(7.k) & \text{Clés} & \mathbf{k_s}^{(k,l)} & = & \mathbf{K}^{(k,l)}\mathbf{h_s}^{(l-1)}, \forall s \leq t \quad (\in \mathbb{R}^{d_{\text{kv}}}) \\
(7.v) & \text{Valeurs} & \mathbf{v_s}^{(k,l)} & = & \mathbf{V}^{(k,l)}\mathbf{h_s}^{(l-1)}, \forall s \leq t \quad (\in \mathbb{R}^{d_{\text{kv}}}) \\
(7.a) & \text{Attention} & \alpha_s^{(k,l)} & = & \text{softmax}(\frac{1}{\sqrt{d_{\text{kv}}}}, \mathbf{o_t})_s, \forall s \leq t \quad (\in [0,1]) \\
 & & \text{avec } o_{ts} & = & \mathbf{q_t}^{(k,l)^T}\mathbf{k_s}^{(k,l)}, \forall s \leq t \\
(7.o) & \text{Sortie} & \mathbf{g_t}^{(k,l)} & = & \sum_{s \leq t} \alpha_s^{(k,l)} \mathbf{v_s}^{(k,l)} \quad (\in \mathbb{R}^{d_{\text{kv}}})
\end{array}
\tag{7}
$$

Les trois premières étapes calculent des projections en dimension d_{kv} des vecteurs de la couche précédente $(l-1)$, appelées respectivement **requête** (*query*), **clé** (*key*) et **valeur** (*value*). Le produit scalaire o_{ts} calcule une similarité entre la requête en position t et les clés à toutes les positions précédant t (t compris). Ces similarités sont normalisées par l'opérateur softmax qui les transforme en **coefficients d'attention** dans $[0, 1]$. La dernière étape combine linéairement les **valeurs** pour calculer le vecteur de sortie.

Chaque couche comprenant plusieurs têtes, il reste à agréger leurs résultats. Deux opérations élémentaires entrent en jeu. La première est une transformation réalisée par un perceptron multi-couche selon :

$$
\left\{
\begin{array}{rcl}
\mathbf{f}_t^{(l)} & = & \phi(\mathbf{W_{if}}^{(l)}\mathbf{g}_t^{(l)} + \mathbf{b_{if}}) \in \mathbb{R}^{d_{\text{ff}}}, \text{ avec } \phi() \text{ une fonction non-linéaire.} \\
\mathbf{h}_t^{(l)} & = & \mathbf{W}_{fo}\mathbf{f}_t^{(l)} + \mathbf{b_{fo}} \in \mathbb{R}^{d_{\text{md}}}.
\end{array}
\right.
\tag{8}
$$

L'entrée de ce perceptron $\mathbf{g}_t^{(l)}$ est la concaténation des sorties des K têtes, à laquelle on ajoute la sortie de la couche précédente : $\mathbf{g}_t^{(l)} = [\mathbf{g}_t^{(1,l)}; \dots; \mathbf{g}_t^{(K,l)}] + \mathbf{h}_t^{(l-1)}$. Ajouter la sortie de la couche précédente sert plusieurs buts : (a) fournir aux gradients un chemin direct depuis les couches hautes vers les couches basses ; (b) assurer que $\mathbf{h_t}^{(l-1)}$ et $\mathbf{h_t}^{(l)}$ restent proches et que chaque mot conserve ainsi ses singularités, indépendemment de l'influence de son contexte. Une conséquence est que les deux termes doivent avoir les mêmes dimensions, ce qui implique $K \times d_{\text{kv}} = d_{\text{md}}$. Une implantation typique de cette étape de propagation avant projette $\mathbf{g}_t^{(l)}$ via $\mathbf{W_{if}}^{(l)}$ dans un vecteur de dimension $d_{\text{ff}} \gg d_{\text{md}}$, par exemple $d_{\text{ff}} = 4d_{\text{md}}$; la non-linéarité de la couche cachée utilise la fonction $\phi() = \text{ReLU}$ (*Rectified Linear Unit*).

La seconde opération élémentaire réalise une **normalisation des sorties**, de manière que les valeurs d'entrées et de sorties restent commensurables. Au final, chaque couche effectue donc une recombinaison de la représentation courante à la position t afin qu'elle intègre l'influence des mots qui la précèdent. Contrairement au modèle récurrent où l'influence des mots du contexte se calcule de proche

en proche, dans ce modèle aucune position n'est privilégiée, et chaque tête d'attention, dans chaque couche, est susceptible d'identifier dans le contexte passé les positions qui sont les plus significatives pour la position courante, à travers les coefficients d'attention $\boldsymbol{\alpha}$.

3.1.2 Conditions limites : les couches 0 et L

Il reste à expliciter les entrées et sorties de ce système. Pour l'entrée $\mathbf{h}_1^{(0)}$, on peut choisir d'utiliser soit des représentations *one-hot* (cf. section 2.3.1), soit des représentations non-contextuelles calculées par un modèle skip-gram [Mikolov et al., 2013] ou des modèles équivalents.

Les représentations lexicales seules ne sont toutefois pas suffisantes. En effet, les équations ((7).[q-v]) ne font aucune distinction entre les indices s, qu'ils soient proches ou éloignés de la position courante t. Ceci illustre le potentiel des Transformers à prendre en compte des influences distantes, mieux que ne le font les réseaux récurrents. Il est pourtant utile (surtout pour les versions décrites ci-dessous) de réintroduire la notion de **position dans la séquence**, par exemple en encodant chaque indice s par un vecteur $\mathbf{p}_s$ en dimension d_{md}, qui est ajouté à la représentation lexicale. Cet **encodage positionnel** sera soit appris [Shaw et al., 2018, Wang and Chen, 2020], soit calculé par une fonction déterministe définie dans [Vaswani et al., 2017] par :

$$\begin{cases} p_s[2i] = \sin(t/10000^{2i/d}) \\ p_s[2i+1] = \cos(t/10000^{2i/d}) \end{cases}$$

Cette formulation présente plusieurs avantages : (a) elle permet d'encoder des indices positionnels arbitrairement grands, tout en conservant les valeurs de chaque composante $p_s[i]$ de la représentation dans un intervalle entre -1 et 1 ; (b) elle conduit à associer des représentations proches à des mots proches dans la phrase [5]. Des travaux récents [Press et al., 2022] proposent d'utiliser des positions relatives et de les intégrer directement au niveau du calcul de l'attention dans l'équation (7), qui intègre un terme qui fera directement dépendre l'attention de l'éloignement entre deux mots (et ce à chaque couche du transformer).

Quant à la sortie de la dernière couche $\mathbf{h}_t^{(L)}$, elle servira à calculer la probabilité du mot à la position $t+1$ et implique les mêmes étapes de calcul que pour le modèle neuronal standard (équation (3)) : transformation linéaire dans un espace de dimension $|V|$ pour obtenir les logits qui seront finalement normalisés en une distribution de probabilité.

5. Pour deux positions t et $t+k$ on a $\sin((t+k)/10000^{2i/d}) - \sin(t/10000^{2i/d}) \approx frack10000^{2i/d}\cos(t/10000^{2i/d})$.

3.2 Le Transformer causal comme pur modèle de langue

La présentation ci-dessus formule le calcul de $\mathbf{h}_t^{(l)}$ comme une opération séquentielle : on calcule $\mathbf{h}_1^{(l)}, l = 1 \dots L$, puis $\mathbf{h}_2^{(l)}, l = 1 \dots L$ dans le contexte de $\mathbf{h}_1^{(l)}, l = 1 \dots L$, etc. C'est la méthode la plus naturelle et computationnellement la plus efficace dans un contexte de modèle de langue, puisque la représentation de chaque mot n'est calculée qu'une fois. Ce modèle est qualifié dans la littérature **d'auto-attentionnel** (car le contexte est constitué des mots précédents dans la même séquence) et de **causal** (la représentation de chaque mot ne dépend que des mots qui le précèdent dans la séquence). Elle est utilisée en particulier dans les architectures GPT-2 et GPT-3 [Radford et al., 2019, Brown et al., 2020]. Une variante **non causale** recalcule toutes les représentations à chaque instant, soit d'abord $\mathbf{h}_1^{(l)}, l = 1 \dots L$, puis $\{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, l = 1 \dots L\}$: cela signifie, par exemple, que la représentation $\mathbf{h}_1^{(l)}$ évolue au cours du temps, intégrant le contexte des mots à sa droite au fur et à mesure de leur découverte. Cette variante, parfois appelée modèle de langue « préfixe » (*prefix language model*) demande plus de calculs, mais semble donner de meilleurs résultats (selon [Raffel et al., 2020]).

Comme les autres modèles de langue, ces architectures permettent de générer des textes de proche en proche, en échantillonnant à chaque instant le prochain mot w_t selon $P(w_t|\mathbf{h}_{t-1})$. Entrainées à très large échelle, elles permettent d'engendrer des énoncés qui sont souvent grammaticalement corrects, voire présentent une forme de cohérence discursive. Ces deux propriétés suggèrent que l'architecture Transformer est capable de modéliser des dépendances entre des mots même très éloignés dans la phrase, voire dans des phrases différentes (lorsque le contexte est suffisamment étendu). Il est également possible de débuter des textes et laisser le modèle poursuivre la génération. Cette forme d'initialisation du texte est qualifiée dans la littérature d'amorçage (*priming*) ou de *prompting*, lorsque l'amorce contient une requête. Il s'avère qu'avec un choix judicieux de ces « amorces », un modèle de langue peut accomplir des tâches multiples : lorsque le prompt est composé d'un texte et de questions de compréhension, le modèle engendre des réponses, lorsque le prompt est une phrase en français, le modèle engendre sa traduction, etc. Il est de surcroit possible, au prix d'un court apprentissage, ou en présentant une poignée d'exemples, d'améliorer ce comportement spontané [Radford et al., 2019]. Ces propriétés remarquables, encore mal comprises, ont permis de positionner les modèles de langue comme des outils de base pour aborder un grand nombre de problèmes de traitement des langues.

3.3 Les Transformers comme représentations : BERT et ses clones

Comme pour les modèles récurrents, l'utilisation des Transformers comme purs extracteurs de représentations contextualisées s'est avérée extrêmement puissante et effective. Proposé initialement dans [Devlin et al., 2019], le modèle BERT est essentiellement un Transformer *non-séquentiel et non-causal*. Ainsi, les calculs de mise à jour de l'équation (7) s'appliquent simultanément **à toutes les positions**

Amorce / Prompt	Génération automatique
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.
Article: Amina Ali Qassim is sitting with her youngest grandchild on her lap, wiping away tears with her headscarf. Only a few months old, this is the baby girl whose ears she desperately tried to cover the night the aerial bombardment started. She lay awake, she says, in a village mosque on the Yemeni island of Birim, counting explosions as the baby cried. It could have been worse though. They could have still been in their house when the first missile landed. "Our neighbor shouted to my husband 'you have to leave, they're coming.' And we just ran. As soon as we left the house, the first missile fell right by it and then a second on it. It burned everything to the ground," Qassim tells us (...) **DL;TR**	Yemen is in the middle of a civil war. Saudi Arabia is leading the coalition bombing campaign. It's been bombing Yemen for more than two months now.
Question: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat? Answer:	dry palms
The Gold Coast Hotel & Casino is a hotel and casino located in Paradise, Nevada. This locals' casino is owned and operated by Boyd Gaming. The Gold Coast is located one mile (~ 1.6km) west of the Las Vegas Strip on West Flamingo Road. It is located across the street from the Palms Casino Resort and the Rio All Suite Hotel and Casino. Question: The Gold Coast is a budget-friendly casino. True, False, or Neither?	Neither

FIGURE 4 – Amorçage d'un modèle de langue. En utilisant différentes stratégies d'amorçage, un même modèle peut être utilisé pour effectuer des tâches multiples. De haut en bas, la génération automatique consiste à continuer un texte, à produire un résumé, à répondre à une question, à vérifier l'implication entre deux phrases, etc. Les extraits sont issus de [Radford et al., 2019, Brown et al., 2020]

de la séquence d'entrée via des calculs matriciels, et les représentations contextuelles de chaque mot intègrent à la fois ses contextes gauche et droit.

Apprendre ce modèle s'avère un peu plus délicat que les modèles directionnels causaux : la solution retenue consiste à bruiter un texte en masquant aléatoirement des mots ; si w_t est masqué, on utilise la représentation $\mathbf{h}_t^{(L)}$ pour le prédire et reconstruire le texte. En maximisant la log-probabilité des mots masqués, il devient possible d'estimer les paramètres du modèle (voir Figure 5).

Entrée	Tous les êtres humains [MASK] libres et égaux en dignité et en droits. [MASK] sont doués de raison et de conscience et doivent [MASK] les uns envers les autres dans un esprit de [MASK].
Sortie	Tous les êtres humains **naissent** libres et égaux en dignité et en droits. **Ils** sont doués de raison et de conscience et doivent **agir** les uns envers les autres dans un esprit de fraternité.

FIGURE 5 – Entrainement du modèle BERT par masquage aléatoire. Les paramètres du modèle sont ajustés pour maximiser la probabilité de reconstituer les mots masqués (en gras dans la sortie).

Comme pour les modèles de langue, cet apprentissage ne requiert pas d'annotation et peut être effectué sur des très grands corpus. BERT fournit ainsi des représentations lexicales contextualisées « pré-entraînées » qui peuvent être utilisées en entrée de n'importe quel système de traitement automatique [Devlin

et al., 2019, Liu et al., 2019]. D'autres méthodes pour construire des tâches de débruitage ont également été proposées : masquage de groupes de mots, masquage partiel de mots, suppressions de mots, permutation aléatoire de l'ordre des mots, etc. Elles sont discutées par exemple dans [Raffel et al., 2020].

Du fait de leurs performances, ces modèles sont devenus centraux en TAL [Smith, 2020] et ont rapidement été portés à de multiples langues, en particulier le français [Martin et al., 2020, Le et al., 2020], le japonais [Kikuta, 2019], l'allemand, le néerlandais [de Vries et al., 2019], l'espagnol [Cañete et al., 2020], le finnois [Virtanen et al., 2019], l'estonien [Tanvir et al., 2020] etc. En parallèle, des versions spécialisées à des sous-genres textuels comme les brevets [Lee and Hsiang, 2020], les textes scientifiques [Beltagy et al., 2019], les tweets [Barbieri et al., 2020] ou à des sous-domaines de spécialité comme les domaines de la biologie-santé [Lee et al., 2020] ou de la physique nucléaire [Jain and Ganesamoorty, 2020] ont également été développés. Une riche littérature s'est également penchée sur l'analyse du comportement empirique des Transformers, visant en particulier à analyser les représentations internes $\{\mathbf{h}_t^l, t = 1 \dots T, l = 1 \dots L\}$ en relation avec les concepts de la linguistique ; ou à utiliser les matrices d'attention comme une explication des décisions du système. Une étude bibliographique récente de cette littérature « Bertologique » ne recense pas moins de 110 références [Rogers et al., 2020].

3.4 Les calculs du Transformer

La place grandissante des architectures Transformer en traitement des langues rend nécessaire de s'intéresser de plus près aux calculs réalisés par ces algorithmes et à leur coût computationnel, dans un contexte où l'empreinte énergétique des algorithmes d'Intelligence Artificielle devient une préoccupation majeure [Strubell et al., 2019, Henderson et al., 2020]. Une première observation est que, contrairement aux réseaux récurrents, les calculs du Transformer sont faciles à paralléliser. En particulier, le calcul de la représentation d'un mot demande de connaître les mots voisins, mais pas de disposer la représentation profonde de sortie associée. Ces représentations peuvent alors toutes être calculées simultanément en implantant l'équation (7) par des calculs matriciels.

Le tableau 1 donne des indications relatives à leur taille et au nombre de paramètres pour quelques modèles récents, dont le nombre de paramètres à estimer se compte aujourd'hui en dizaines de milliards.

TABLE 1 – Dimensions de modèles Transformer typiques. Le nombre de paramètres des représentations lexicales (L) et des paramètres internes du transformer (I) sont distinguées. Les notations k,m,b représentent respectivement des milliers, millions et milliards.

$\|V\|$	T	K	L	d_{md}	d_{kv}	d_{ff}	Params (L)	Params (I)
32k	512	8	6	512	64	2048	32,8m	49,9m

$\|V\|$	T	K	L	d_{md}	d_{kv}	d_{ff}	Params (L)	Params (I)
32k	512	12	12	768	64	3072	49,2m	127m
32k	512	16	24	1024	64	4096	65,5m	342m
32k	512	32	24	1024	128	16384	65,5m	2,38b
32k	512	128	24	1024	128	65536	65,5m	28,7b

Une dernière dimension importante pour les calculs de complexité est la longueur des séquences T à traiter, qui détermine la dimension globale de l'entrée et de la sortie de chaque couche ($T \times d_{\text{md}}$). T contient typiquement plusieurs centaines de mots, voire quelques milliers (2048 pour GPT-3). Pendant l'entrainement il est nécessaire de conserver en mémoire les valeurs de toutes les couches qui sont utilisées pour le calcul du gradient. Pour optimiser les calculs et stabiliser le gradient, on traite simultanément des blocs (*batchs*) de B séquences, qui correspondent à des tenseurs de dimension $B \times T \times d_{\text{md}}$, dont le traitement matriciel est effectué efficacement sur des cartes GPU.

La complexité computationnelle du calcul du Transformer est dominée par l'évaluation des matrices d'attention dans l'équation (7). Ce calcul est une opération linéaire en d_{md}, mais **quadratique en** T : pour chacune des T positions, il faut calculer la similarité avec toutes les autres positions, afin d'en dériver les attentions $\boldsymbol{\alpha}$, puis les T valeurs de sortie. Pour réduire la complexité, plusieurs voies sont explorées dans la littérature. Il est ainsi possible de restreindre les calculs d'attention au **voisinage** $N(t)$ **du mot** w_t, en imposant que les mots à l'extérieur de $N(t)$ aient une attention $\alpha_t(s)$ nulle ; ces mots influencent toutefois w_t indirectement en influençant ses voisins (ou les voisins de ses voisins) au fil des différentes couches de calcul. En choisissant des voisinages $N(t)$ de taille S fixe et petite devant T, la complexité du calcul devient linéaire en T. On peut ainsi contraindre $N(t)$ à ne contenir que les mots proches dans la séquence, ou proches syntaxiquement, ou bien choisis aléatoirement. L'important est que pour presque tous les mots, $|N(t)|$ soit petit et que, pour quelques positions, $|N(t)|$ englobe toute la séquence. D'autres approches se focalisent sur des approximations efficaces des produits scalaires (équation (7).a.

Une étude récente des implantations efficaces du modèle Transformer est dans [Tay et al., 2020] ; pour ce qui concerne l'optimisation de la mémoire, on pourra se reporter à Dans la mesure où accroitre les volumes de données traitées permet d'améliorer les performances pour de nombreuses tâches [Kaplan et al., 2020], il reste tentant d'entrainer des modèles toujours plus grands [Fedus et al., 2021], ce qui continue de poser de redoutables défis computationnels aussi bien lors de l'apprentissage que lors de l'exploitation. Cette tendance a également pour effet d'exclure nombre d'équipes de recherche de ces travaux, faute d'avoir accès aux infrastructures de calcul nécessaires.

3.5 Conclusion

Les modèles de langue implantés dans les architectures Transformer combinent tous les avantages des architectures neuronales : ils permettent d'apprendre à la fois des modèles prédictifs capables de prendre en compte des dépendances à longue distance et des représentations contextuelles riches pour les unités modélisées, susceptibles d'être pré-entrainées, puis utilisées pour des multiples tâches de traitement linguistique. Ils donnent lieu à des implantations efficaces [Wolf et al., 2020], qui ont également été adaptées pour d'autres types de données structurées : des séquences acoustiques pour la modélisation de la parole [Baevski et al., 2020], des images pour la vision artificielle [Qi et al., 2020], voire des séquences d'images [Sun et al., 2019]. À l'instar des autres modèles de langue neuronaux, leur fonctionnement reste difficile à contrôler : si certaines régularités sont apprises presque parfaitement, d'autres le sont de manière moins satisfaisante, sans que l'on puisse prédire ou expliquer les raisons de ces échecs.

3.6 À retenir

— Le modèle Transformer s'appuie sur le mécanisme d'attention pour calculer des représentations latentes intégrant un contexte étendu sans utiliser de récurrence.
— L'apprentissage du Transformer peut être effectué en parallèle sur tous les mots d'une séquence.
— Il peut servir comme modèle de langue (version causale) mais également pour apprendre des représentations lexicales contextualisées (version non causale).
— Des implantations efficaces, ainsi que de multiples modèles pré-appris sont aujourd'hui disponibles pour de nombreuses langues et domaines.

4 Vers le multilinguisme

4.1 La traduction automatique neuronale : génération conditionnelle de textes

4.1.1 Le modèle encodeur-décodeur simple

Le modèle Transformer présenté ci-dessus comme un modèle de langue est initialement introduit dans un contexte de traduction automatique (TA) [Vaswani et al., 2017]. Cette application correspond formellement à la génération (en langue cible) d'une phrase $\mathbf{e}$ traduisant la phrase source $\mathbf{f}$ en entrée. Exprimé comme une décision probabiliste, ce problème correspond à trouver :

$$\mathbf{e}^* = \text{argmax}_{\mathbf{e}}\, P(\mathbf{e}|\mathbf{f}) = \text{argmax}_{\mathbf{e}} \prod_t P(e_t|\mathbf{f}, \mathbf{e}_{<\mathbf{t}}). \tag{9}$$

Cette formalisation demande à nouveau de définir une distribution de probabilité

sur un ensemble de phrases (cf. l'équation (1)), à ceci près que cette distribution est **conditionnelle à la phrase d'entrée f**. Le modèle Transformer fournit un tel modèle en prolongeant les architectures neuronales encodeur-decodeur proposées pour la TA par [Cho et al., 2014a, Bahdanau et al., 2015]. Ces architectures formalisent le problème précédent en deux étapes de calcul :

(a) . le calcul d'une représentation numérique pour $\mathbf{f}$ (encodage), prenant la forme d'une suite de vecteurs numériques ;

(b) . le décodage itératif de la traduction sélectionnant à chaque étape le mot e_t le plus probable sachant l'encodage $[\mathbf{g}_1^{(l)}, \ldots, \mathbf{g}_I^{(l)}], l = 1 \ldots L$ de la phrase source $\mathbf{f}$, ainsi que les mots cibles déjà produits $\mathbf{e}_{<t}$, représentés comme précédemment par $[\mathbf{h}_1^{(l)}, \ldots, \mathbf{h}_{t-1}^{(l)}], l = 1 \ldots L$.

Les premiers systèmes de TA neuronale réalisent ces deux étapes au moyen de réseaux récurrents (section 2.3.2). Dans une architecture à base de Transformer, l'étape (a) est réalisée par un encodeur non-causal (section 3.3) et l'étape (b) est réalisée par un décodeur causal (section 3.2). Durant cette seconde étape, il faut combiner la double dépendance en $\mathbf{f}$ et $\mathbf{e}_{<t}$, puisque la prédiction du prochain mot cible est influencée simultanément par ces deux séquences (cf. l'équation (9)). Cette interaction est opérée par l'adjonction d'une sous-couche supplémentaire dite **cross-attentionnelle** au sein du décodeur. Les calculs qu'elle réalise reprennent ceux de l'équation (7), en utilisant $\mathbf{h}_t^{(l)} \ldots \mathbf{h}_t^{(l)}$ pour la requête, et $\mathbf{g}_1^{(L)}, \ldots, \mathbf{g}_J^{(L)}$ pour les clés et valeurs. De cette manière, le vecteur de contexte de chaque mot cible intègre non seulement les mots précédents, mais **également l'ensemble des mots de la phrase source**, tels qu'ils sont représentés à la dernière couche (L) de l'encodeur. Comme précédemment, le vecteur en dernière couche du décodeur est projeté dans un espace $|V|$-dimensionnel, puis normalisé par la fonction softmax pour fournir la distribution souhaitée $P(\mathbf{e}_t|\mathbf{f}, \mathbf{e}_{<t})$.

4.1.2 Difficultés de la TA

Apprendre à traduire avec des corpus parallèles

L'apprentissage de modèles conditionnels est en tout point similaire à l'apprentissage de modèles de langue et consiste à maximiser la log-probabilité des séquences d'apprentissage, qui se décompose comme une somme de termes de la même forme que dans l'équation (4). Pour réaliser ce calcul, il faut connaitre à la fois les mots de la phrase source et de la phrase cible, donc disposer de grands *corpus parallèles* appariant des phrases et leur traduction. De telles ressources sont aujourd'hui publiquement disponibles en masse sur des sites spécialisés ou auprès d'agences de diffusion de ressources comme ELDA ou le Linguistic Data Consortium. On trouve également des corpus très variés sur les sites spécialisés tels que OPUS [Tiedemann, 2012].

Difficultés de la génération automatique

Une fois cet apprentissage terminé (ce qui peut prendre plusieurs jours selon

la taille des données parallèles, qui se comptent parfois en millions de lignes, et la puissance des calculateurs disponibles), le modèle Transformer est prêt à traduire. La traduction se déroule incrémentalement, mot après mot, de manière gloutonne (*greedy*) et pose les mêmes problèmes difficiles que la génération de texte inconditionnelle. Il apparait d'une part que choisir à chaque pas de temps le meilleur mot est une stratégie risquée : chaque erreur dans le passé construit des représentations internes incorrectes ou simplement inhabituelles, qui peuvent à leur tour engendrer d'autres erreurs. Ce problème est connu comme le problème du **biais d'exposition** [Bengio et al., 2015] et demande d'utiliser des stratégies de recherche de l'argmax (équation (9)) plus sophistiquées, par exemple la **recherche en faisceau** (*beam search*). Une manière plus radicale de contourner le problème consiste à abandonner la dépendance entre e_t et les mots qui le précèdent $e_{<t}$ pour prédire **simultanément en parallèle** tous les mots de la sortie, ce qui a également pour effet d'accélérer le décodage. Des contraintes globales sur les positions relatives des mots doivent toutefois s'appliquer pour garantir que la phrase cible restera bien formée [Gu et al., 2018].

Deux difficultés supplémentaires sont directement liées au cadre de traduction automatique. En TA, il est en effet nécessaire de traduire **l'intégralité de la phase source** (chaque mot devant être traduit une fois et une seule), sans introduire **d'information supplémentaire**. Pourtant ces deux contraintes ne sont pas explicitement formulées dans l'équation (9) : pour s'assurer que la phrase cible possède une longueur compatible avec celle de la source, et traduit effectivement tous les mots de l'entrée, l'algorithme de recherche doit inclure des heuristiques supplémentaires : [Johnson et al., 2017] présente les heuristiques les plus communément utilisées.

4.2 Représentations multilingues, traduction multilingues

Un bénéfice des représentations numériques est qu'elles représentent de la même manière des mots de langues différentes. Il est alors possible, pour autant qu'on dispose d'un répertoire partagé d'unités, d'utiliser les mêmes encodeurs et décodeurs pour traiter plusieurs langues. La façon la plus simple de procéder est de mettre en œuvre la même approche que pour BERT (section 3.3) et de présenter au système des phrases rédigées en plusieurs langues [Conneau and Lample, 2019] : cette approche est utilisée pour les modèles mBERT et XLM.

Lorsqu'il s'agit seulement de calculer des représentations, cette stratégie permet d'apprendre des **représentations contextuelles multilingues** qui rapprochent dans un même espace les représentations d'unités (des mots, des phrases) qui sont des traductions mutuelles. L'apprentissage de représentations multilingues rend indiscernables (pour le réseau de neurones) des phrases écrites dans des langues différentes. Elle permet ainsi de transférer des modèles de traitement et des applications depuis une langue riche en ressources vers des langues pour lesquelles les ressources n'existent pas. Prenons l'exemple d'un système **d'analyse de sentiments**, qui vise à associer des commentaires textuels sur un site marchand avec des notes de satisfaction. Supposons que l'on dispose d'exemples

d'apprentissage pour la langue A, mais pas pour la langue B. Si l'on apprend à prédire la note à partir des représentations multilingues des textes en langue A, alors on pourra également prédire la note des textes en langue B **sans avoir jamais observé aucun exemple d'apprentissage** associant un texte en langue B avec son évaluation.

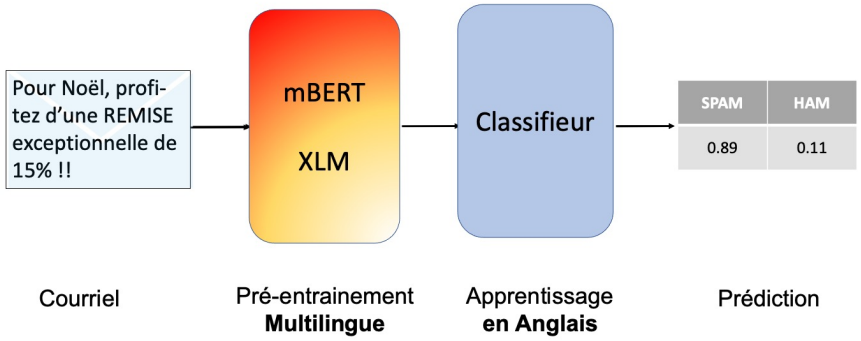

FIGURE 6 – Architectures pour le filtrage de pourriels. La seconde architecture utilise des représentations pré-entrainées et le troisième le transfert cross-lingue, ce qui permet de classer les courriels rédigés en français à partir d'un apprentissage réalisé sur des courriels rédigés en anglais.

La construction de représentations multilingues est donc un enjeu majeur pour élargir le spectre des langues couvertes par les technologies linguistiques. De nombreuses approches ont été également proposées pour entrainer des représentations multilingues non-contextuelles, ou encore pour adapter à un domaine des représentations existantes. On se reportera à A pour une analyse de la littérature récente sur ces questions.

Notons enfin que l'architecture encodeur-décodeur peut également être utilisée pour calculer des représentations **monolingues** : il suffira pour cela de présenter à l'encodeur des entrées bruitées de phrases correctes, que le système devra apprendre à reconstituer (débruiter). Ici encore, nul besoin d'annotation, puisqu'il suffira juste de spécifier les fonctions de bruits idoines : par exemple masquer un ou plusieurs mots, remplacer un mot par un mot proche, permuter l'ordre des mots sont des transformations typiques et faciles à utiliser pour générer des données parallèles artificielles. Cette architecture, introduite par [Lewis et al., 2020] sous le nom de BART, présente par rapport à BERT l'avantage d'un apprentissage accéléré (plus de mots sont bruités dans l'encodeur). Au prix d'une étape d'affinage (*fine tuning*), BART peut également être utilisé comme modèle

de génération puisque le décodeur est non-causal : une application possible est le résumé automatique. Notons enfin que comme BERT, BART peut être entrainé de manière multilingue, calculant simultanément des représentations multilingues et une traduction automatique.

4.3 Un modèle pour traduire toutes les langues

La **traduction multilingue** combine les approches décrites dans les sections précédentes : l'utilisation d'une architecture encodeur-décodeur, avec génération conditionnelle de textes ; la combinaison de couples de phrases associant entrée et sortie pour de multiples paires de langues. Cette idée, initialement proposée dans [Firat et al., 2016, Ha et al., 2016] et récemment utilisée à grande échelle dans [Aharoni et al., 2019, Fan et al., 2021] ouvre de nouvelles perspectives : un seul système à maintenir là où il faudrait $O(N^2)$ systèmes pour traiter N langues, avec en plus la possibilité de produire des traductions pour des langues pour lesquelles aucune donnée n'est observée (de nouveau par un transfert entre langues, qui opère ici à la fois dans l'encodeur et dans le décodeur).

Cette approche n'est pas sans difficulté, en particulier du point de vue du recueil et de l'équilibrage des données parallèles d'apprentissage, ainsi que de celui de la prise en charge de systèmes linguistiques très différents, qui peuvent par exemple utiliser des systèmes d'écriture différents, ou bien manipuler des structures sous-jacentes (les notions de mot ou de syntagmes) divergentes. Un prétraitement indispensable est l'apprentissage d'outils de segmentation (en mots et sous-mots, cf. section 2.4) du texte sur des corpus multilingues, qui assurent que toutes les entrées-sorties du système utilisent le même vocabulaire d'unités.

4.4 La traduction : une tâche modèle

4.4.1 Extension du domaine de la traduction

Le passage du modèle inconditionnel (section 3) au modèle conditionnel (section 4.1) illustre la flexibilité des représentations continues manipulées par les réseaux neuronaux : en ajoutant un mécanisme d'attention croisée entre deux Transformers, on peut simultanément encoder deux séquences de mots dans un vecteur $\mathbf{h}$, à partir duquel prédire le prochain mot. Cette technique s'étend à l'encodage de contextes étendus permettant de modéliser des tâches ou des scénarios d'usage plus complexes. Il est ainsi possible de modéliser les scénarios de **traduction multi-source**, correspondant à la génération d'une phrase cible depuis deux phrases sources $\mathbf{f_1}$ et $\mathbf{f_2}$. Pour modéliser la distribution $P(\mathbf{e}|\mathbf{f_1}, \mathbf{f_2})$, on pourra par exemple distribuer l'attention cross-lingue sur la concaténation des deux encodages. Autre illustration de cette plasticité, la **traduction au niveau document**, qui vise à intégrer des dépendances distantes excédant le niveau de la phrase. Un exemple typique concerne les références pronominales, comme dans la traduction depuis l'anglais de : *This bike is broken. It needs a fix.* Pour cet exemple, la génération d'un pronom sujet de la deuxième phrase (*il* ou *elle*) demande de savoir comment *bike* a été traduit à la phrase précédente (*vélo*

ou *bicyclette*). En encodant des contextes intégrant plusieurs phrases précédentes, il devient possible de traiter ces difficultés [Maruf et al., 2021].

4.4.2 Traduction monolingue, traduction multi-modale

La traduction automatique est un exemple extrême de tâche de transformation d'une séquence de mots, correspondant à un changement de langue, tout en préservant la sémantique. Ce cadre s'applique à un grand nombre d'autres tâches **monolingues** : par exemple la correction orthographique peut être traitée comme une « traduction » entre un énoncé bruité et sa correction, cadre qui englobe également la normalisation orthographique (par exemple : passer de la langue SMS au français standard). Simplification, génération de reformulations et de paraphrases, transfert de style (par exemple d'un style formel à un style plus relâché), résumé automatique [Liu et al., 2018] sont autant d'autres exemples de ces traductions monolingues : pour autant que l'on disposera de couples (entrée, sortie) pour l'apprentissage, il sera possible d'utiliser des architectures Transformer.

Cette architecture encodeur-décodeur se généralise également à d'autres modalités. En considérant des couples associant des enregistrements vocaux à leur transcription, il est possible, si l'on sait encoder le signal de parole, de l'appliquer pour passer de la voix au texte [Dong et al., 2018, Karita et al., 2019, Gulati et al., 2020], voire, lorsque les enregistrements et les textes sont dans des langues différentes, de traduire directement la parole [Bérard et al., 2016]. Des approches similaires s'intéressent à la reconnaissance d'images [Dosovitskiy et al., 2021] ou la génération de descriptions à partir d'images [Lu et al., 2019]. L'application des Transformers à ces autres modalités n'en est qu'à ses débuts et il est attendu qu'elle se développera, aussi bien pour apprendre des modèles de génération que pour apprendre des représentations combinant plusieurs modalités.

4.5 À retenir

- L'architecture Transformer s'étend au cadre de génération conditionnelle, avec un encodeur et un décodeur interdépendants.
- Ce cadre est devenu le cadre de référence pour la traduction automatique neuronale.
- Alimenté par des entrées (et sorties) en plusieurs langues, il construit des représentations multilingues utilisables par transfert cross-lingue dans de nombreuses applications.
- Par analogie avec la traduction, ce modèle est utilisable pour de nombreuses tâches monolingues comme pour des tâches impliquant d'autres modalités (parole, image, vidéo).

5 Conclusion

L'architecture Transformer, dans sa version conditionnelle (section 4) comme dans sa version inconditionnelle (section 3), s'est rapidement imposée comme un composant essentiel de tous les systèmes de traitement linguistique, et a souvent permis des améliorations considérables des performances de ces systèmes. Cette architecture permet d'entraîner, avec de très grandes masses de données non-annotées, des représentations contextuelles qui sont utilisables pour une vaste gamme d'applications, et de transferer la connaissance apprise entre tâches, domaines et langues. En conséquence, elle offre une réponse opérationnelle face au manque de données annotées qui seraient nécessaires pour réaliser un apprentissage supervisé. Elle permet également d'apprendre des modèles de génération de mots capables d'engendrer des textes cohérents, et qui permettent également, au prix de reformulations élémentaires, de traiter un grand nombre de tâches : détection de sentiment, implication textuelle, réponse à des questions, résumé, traduction, etc. Des extensions multilingues et multimodales de ces architectures permettent de construire des modèles à partir de données hétérogènes, élargissant ainsi la gamme des applications possibles. Elles fournissent enfin un cadre conceptuel partagé pour de nombreuses communautés de chercheurs et de développeurs, facilitant les échanges interdisciplinaires et accélérant la diffusion d'implantations efficaces et le partage de modèles.

Doit-on pour autant penser que cette architecture constitue un aboutissement pour les technologies linguistiques ? Plusieurs limitations de ces modèles sont pointées dans la littérature récente, suggérant autant de pistes de recherche pour des travaux futurs. Une première limitation est que ces modèles n'intègrent pas de connaissance linguistique (sur la structure des mots, des phrases, ou des annotations), ce qui les rend impropres à reproduire le comportement systématique qui est attendu lorsque l'on traite des phénomènes réguliers, comme par exemple l'accord grammatical, ou les phénomènes de coréférence. Bien que possible, cette intégration de connaissances linguistiques va à l'encontre de l'augmentation de la taille des données et du nombre de langues prises en compte, et ne constitue pas une piste de recherche très active. De même, la seule connaissance du monde fournie aux Transformers est celle qui est disponible dans les textes qui servent à l'apprentissage. Bien que ces modèles soient capables de mémoriser et restituer un grand nombre de ces connaissances factuelles, elles n'en restent pas moins lacunaires, et leur apprentissage est incertain et non-systématique [Petroni et al., 2019], au point qu'il est impropre de leur prêter la moindre capacité de compréhension des langues [Bender and Koller, 2020]. Pour progresser dans cette direction, la combinaison de modèles statistiques et de graphes de connaissance semble une piste prometteuse [Peters et al., 2019, Bosselut et al., 2019].

Une autre limitation de ces architectures est leur fonctionnement en "boîte noire", qui créé de multiples problèmes d'usage dès lors que ces systèmes sont exploités à très grande échelle. En particulier, il est impossible d'expliquer simplement les décisions prises, car elles ne résultent finalement que de la dynamique particulière de l'entrainement du modèle et de la nature des données

qui ont servi à l'apprentissage. Comme il a été montré à de nombreuses reprises [Bender et al., 2021], ces modèles ont en particulier tendance à amplifier les biais présents dans les données, et peuvent par exemple produire, de manière incontrôlée, des énoncés à caractère sexiste ou raciste. L'impression de cohérence des textes produits automatiquement est également trompeuse, car elle incite à prêter à ces systèmes une forme de compréhension des énoncés qu'ils ne possèdent en aucune manière. Ces travers sont inhérents à tous les modèles purement probabilistes, qui se heurtent aux limitations des données d'apprentissage, qui trop rares, incomplètes, ou biaisées, conduisent à apprendre des systèmes souvent incomplets et incohérents dans leur fonctionnement.

6 Remerciements

Cet article a bénéficié d'une relecture attentive de la part de mes collègues M. Evrard, C. Guinaudeau, S. Pogodalla et J.-P. Haton, ainsi que de discussions avec J. Crowley. Je les en remercie vivement.

7 Glossaire

Affinage ; *Finetuning* : Un modèle entraîné pour une tâche particulière (par exemple un sur une tâche de modèle de langue) peut-être transféré vers une autre tâche en prolongeant l'apprentissage avec d'autres types de données ou d'annotations : c'est l'étape **d'affinage**. Ainsi, les paramètres d'un modèle comme BERT pourront être spécialisés en utilisant quelques exemples d'une tâche d'analyse de sentiments. L'affinage de modèles est une des méthodes pour apprendre par transfert.

Annotation ; *Labeling* : Les données annotées sont nécessaires pour guider l'apprentissage supervisé. Les annotations peuvent porter sur un texte, une phrase, ou encore des mots isolés ; elles peuvent être de nature linguistique (morphologique, syntaxique, sémantique), ou encore représenter la sortie d'une tâche de traitement (par exemple la polarité d'un texte, ou encore l'équivalence sémantique entre deux phrases).

Apprentissage par transfert ; *Transfer Learning* : L'apprentissage par tranfert consiste à apprendre un modèle probabiliste ou neuronal avec des textes annotés pour une tâche, un domaine ou une langue A, puis à l'exploiter pour traiter des textes d'un autre domaine ou d'une autre langue B. Pour le transfert entre tâches ou domaines, on peut utiliser des méthodes de finessage. Pour le transfert entre langues (*cross-lingual transfer*), il est rendu possible par l'utilisation de représentations multilingues, qui représentent des mots de langues différentes dans un même espace, en rapprochant les mots qui sont des traductions mutuelles.

Apprentissage avec peu d'exemples, apprentissage oligo-exemples ; *few-shot learning* : L'apprentissage « oligo-exemples » d'un modèle consiste à l'exploiter pour réaliser des traitement ou des annotations après un entrainement