Décodage guidé par un discriminateur avec le Monte Carlo Tree Search pour la génération de texte contrainte

Antoine Chaffin, Vincent Claveau, Ewa Kijak

▶ **To cite this version:**

Antoine Chaffin, Vincent Claveau, Ewa Kijak. Décodage guidé par un discriminateur avec le Monte Carlo Tree Search pour la génération de texte contrainte. TALN 2022 - 29e conférence sur le Traitement Automatique des Langues Naturelles, Jun 2022, Avignon, France. pp.27-41. hal-03701490

HAL Id: hal-03701490
https://hal.science/hal-03701490

Submitted on 24 Jun 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Décodage guidé par un discriminateur avec le Monte Carlo Tree Search pour la génération de texte contrainte

Antoine Chaffin[1, 2] Vincent Claveau[1] Ewa Kijak[1]
(1) Univ. Rennes – CNRS – IRISA, Rennes, France
(2) IMATAG, Rennes, France
prenom.nom@irisa.fr

RÉSUMÉ

Dans cet article, nous explorons comment contrôler la génération de texte au moment du décodage pour satisfaire certaines contraintes (e.g. être non toxique, transmettre certaines émotions...), sans nécessiter de ré-entrainer le modèle de langue. Pour cela, nous formalisons la génération sous contrainte comme un processus d'exploration d'arbre guidé par un discriminateur qui indique dans quelle mesure la séquence associée respecte la contrainte. Nous proposons plusieurs méthodes originales pour explorer cet arbre de génération, notamment le Monte Carlo Tree Search (MCTS) qui fournit des garanties théoriques sur l'efficacité de la recherche. Au travers d'expériences sur 3 jeux de données et 2 langues, nous montrons que le décodage par MCTS guidé par les discriminateurs permet d'obtenir des résultats à l'état-de-l'art. Nous démontrons également que d'autres méthodes de décodage que nous proposons, basées sur le re-ordonnancement, peuvent être réellement efficaces lorsque la diversité parmi les propositions générées est encouragée.

ABSTRACT

Discriminator-guided decoding with Monte Carlo Tree Search for constrained text generation

In this paper, we explore how to control text generation at decoding time to satisfy certain constraints (eg. being non-toxic, conveying certain emotions...) without fine-tuning the language model. Precisely, we formalize constrained generation as a tree exploration process guided by a discriminator that indicates how well the associated sequence respects the constraint. We propose several original methods to search this generation tree, notably the Monte Carlo Tree Search (MCTS) which provides theoretical guarantees on the search efficiency.Through 3 tasks and 2 languages, we show that discriminator-guided MCTS decoding achieves state-of-the-art results without having to tune the language model. We also demonstrate that other proposed decoding methods based on re-ranking can be really effective when diversity among the generated propositions is encouraged.

MOTS-CLÉS : Génération de texte, génération collaborative, décodage, Monte Carlo Tree Search.

KEYWORDS: Text generation, collaborative generation, decoding, Monte Carlo Tree Search.

1 Introduction et état de l'art

Les modèles de langue génératifs (LM pour *Language Models*) existent depuis longtemps, mais avec l'avènement de l'architecture des transformers (Vaswani *et al.*, 2017) et l'augmentation des capacités de calcul, ils sont maintenant capables de générer des textes longs et bien écrits dans beaucoup de situations. Ces LM, tels GPT-2 et 3 (Radford *et al.*, 2019; Brown *et al.*, 2020), ont été

utilisés avec succès pour diverses applications : aide à la rédaction, résumé, augmentation de données, génération de fausses nouvelles (Kumar *et al.*, 2020; Papanikolaou & Pierleoni, 2020; Zellers *et al.*, 2019). Pourtant, à part l'amorce (*prompt*) initiant la génération, il existe peu d'options pour contrôler le processus de génération pour, par exemple, adopter un certain style, transmettre une émotion spécifique, prévenir la toxicité ou les stéréotypes. Formellement, la génération avec une contrainte c vise à trouver la séquence de tokens $x_{1:T}$ du vocabulaire $\mathcal{V}$ maximisant $p(x_{1:T} \mid c)$.

La plupart des méthodes existantes nécessitent un ré-apprentissage (*fine-tuning*) du LM, afin qu'il apprenne spécifiquement à modéliser cette contrainte qui est alors inhérente au LM. Un LM doit alors être entraîné spécifiquement pour chaque contrainte (qui doit être définie au préalable), ce qui est coûteux voire impossible compte tenu de la taille des LM actuels.

Une autre famille d'approches utilise un discriminateur (classifieur) D pour guider la génération au moment du décodage. Toute contrainte supplémentaire peut alors être définie a posteriori sans avoir à ré-entraîner le LM, seulement en entraînant un autre discriminateur, ce qui est moins coûteux. Les discriminateurs ont été utilisés de différentes manières pour explorer l'espace de recherche (des séquences possibles) lors de la génération.

Ces méthodes existantes ont plusieurs limites. Certaines n'exploitent le discriminateur qu'en fin de génération, ce qui filtre plus que ne guide la génération et peut donc manquer des solutions intéressantes. À l'inverse, les méthodes exploitant D au cours de la génération souffrent d'un manque de vision à long terme de la génération. En effet, les sorties du discriminateur D peuvent n'être fiables pour guider la recherche que lorsque la séquence est (presque) terminée. Dans une stratégie de décodage myope, les erreurs de classification au début des séquences feront dévier le processus de génération. Dans cet article, nous proposons de nouvelles approches exploitant un discriminateur pour guider le décodage vers des textes respectant une contrainte voulue. Elles ne nécessitent que d'accéder aux logits du LM, ce qui les rend utilisables avec n'importe quel LM, y compris ceux n'offrant qu'un accès limité à leurs sorties (par exemple, fournies par l'API de GPT-3).

Précisément, nos principales contributions sont les suivantes :

1. nous proposons une méthode originale de décodage pour la génération contrainte basée sur l'algorithme MCTS (Monte Carlo Tree Search) (Coulom, 2006), que nous appelons Plug and Play Language - Monte Carlo Tree Search (PPL-MCTS), et nous montrons, sur 3 jeux de données et 2 langues, qu'elle donne des résultats à l'état-de-l'art tout en offrant une grande flexibilité ;

2. nous explorons également des méthodes plus simples basées sur le réordonnancement et nous montrons que ce type d'approche, faible en coût de calcul, peut être compétitif si la diversité au sein des propositions à re-ordonner est encouragée ;

3. nous fournissons un code entièrement fonctionnel mettant en œuvre un MCTS pour le texte, permettant le traitement en lots (*batch*), reposant sur la bibliothèque Transformers d'HuggingFace (Wolf *et al.*, 2020).

2 Travaux connexes

Peu de travaux traitent de la génération textuelle sous contrainte. Son but est de trouver la séquence de tokens $x_{1:T}$ qui maximise $p(x_{1:T} \mid c)$, étant donné une contrainte c.

2.1 Méthodes fondées sur un ré-apprentissage

Les Class-conditional language models (CC-LM), comme le modèle CTRL (Conditional Transformer Language) (Keskar *et al.*, 2019), entraînent les poids θ d'un unique modèle neuronal, en ajoutant un code de contrôle au début de chaque séquence d'entraînement qui indique la contrainte (ou la classe) de cette séquence. Entraîné avec différents codes de contrôle, le modèle apprend $p_\theta(x_{1:T} \mid c) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{1:t-1}, c)$. La contrainte peut ensuite être appliquée pendant la génération en ajoutant à l'amorce le code de contrôle voulu. Les codes de contrôle doivent donc être définis au préalable, et le LM doit être entraîné spécifiquement pour chaque ensemble de codes de contrôle, ce qui est difficile avec les très gros LM. Il s'agit d'une limitation importante car la tendance actuelle en matière de génération de texte est l'utilisation d'un grand modèle pré-entraîné qui peut difficilement être affiné (par exemple, GPT-3 ne peut être affinée sans l'accès à de très grandes ressources matérielles).

2.2 Méthodes fondées sur des discriminateurs

Cette famille de méthodes utilise un discriminateur D pour guider la génération au moment du décodage. Le discriminateur modélise explicitement la contrainte en calculant la probabilité $p_D(c \mid x_{1:T})$ que la séquence $x_{1:T}$ satisfasse la contrainte c. Cette probabilité est directement liée à $p(x_{1:T} \mid c)$ par la règle de Bayes :

$$p(x_{1:T} \mid c) \propto p_D(c \mid x_{1:T}) p_\theta(x_{1:T})$$

Les discriminateurs sont moins coûteux à entraîner qu'un LM et toute contrainte supplémentaire peut être définie a posteriori sans avoir à ré-entraîner le LM, seulement en entraînant un autre discriminateur.

Les discriminateurs ont été utilisés de différentes manières pour explorer l'espace de recherche (du prochain token) lors de la génération. Holtzman *et al.* (2018) et Scialom *et al.* (2020) explorent d'abord l'espace par faisceau (*beam search*) pour générer un ensemble de propositions avec une probabilité élevée $p_\theta(x_{1:T})$, puis D est utilisé pour les reclasser selon $p_D(c \mid x)$. La *beam search* peut cependant manquer des séquences à forte probabilité et est biaisée par la probabilité du LM, alors que la meilleure séquence peut n'avoir qu'une probabilité moyenne, mais satisfaire parfaitement la contrainte.

Il peut donc paraître plus approprié de prendre en compte D pendant le décodage plutôt qu'après avoir généré une séquence entière. Dans ce cas, D est utilisé à chaque étape de génération pour obtenir $p_D(c \mid x_{1:t})$ pour chaque token du vocabulaire $\mathcal{V}$, et la fusionner à la vraisemblance $p_\theta(x_{1:t})$ pour choisir le token à émettre.

Afin de réduire le coût de l'utilisation de D sur chaque continuation possible (ie. scorer chaque token du vocabulaire), GeDi (Krause *et al.*, 2020) propose d'utiliser les CC-LM comme discriminateurs génératifs. La méthode repose sur le fait que le CC-LM calcule $p_\theta(x_t \mid x_{1:t-1}, c)$ pour chaque token du vocabulaire en une passe. Cette probabilité peut alors être utilisée pour obtenir $p_\theta(c \mid x_{1:t})$ pour chaque token en utilisant l'équation de Bayes. GeDi se situe donc à l'intersection entre le ré-entrainement d'un LM et l'utilisation d'un discriminateur : il ré-entraîne un petit LM (le CC-LM) pour guider la génération d'un LM plus grand.

Dans le modèle de langue Plug and Play (PPLM) (Dathathri *et al.*, 2020), D est utilisé pour modifier les états cachés d'un LM à base de transformers pré-entraîné vers la classe désirée à chaque étape de génération. PPLM doit donc accéder au LM pour modifier ses états cachés, ce qui est une contrainte

forte. Notre approche, au contraire, ne nécessite que d'accéder aux logits (par exemple, fournis par l'API de GPT-3), cela rend notre approche encore plus "plug and play" que PPLM.

Comme souligné dès l'introduction, un inconvénient commun à toutes ces approches est le manque de vision à long terme de la génération. En effet, les probabilités de D deviennent nécessairement plus significatives au fur et à mesure de la croissance de la séquence et peuvent n'être fiables pour guider la recherche que lorsque la séquence est suffisamment longue ou terminée. Or, les erreurs de classification sur les premiers tokens font dévié le processus de génération vers des solutions moins optimales. L'approche que nous proposons répond en partie à ce problème grâce à l'utilisation de l'algorithme MCTS, ici appliqué à l'exploration de l'espace des générations. Nous le détaillons dans la section suivante.

3 L'approche PPL-MCTS

L'approche que nous proposons utilise un discriminateur pour guider le décodage mais veut apporter une vision à long terme de ce qui est généré. L'espace de recherche de la génération de texte correspond à un arbre : sa racine est l'amorce et l'enfant d'un nœud est la séquence de son père avec l'un des tokens de $\mathcal{V}$ ajouté. Dans notre cas, le but est de trouver le chemin (i.e. la séquence) x, avec $p(x \mid c)$ le plus grand possible sans explorer l'arbre entier en largeur et en profondeur. Cette probabilité peut être calculée par $p_\theta(x) * p_D(c \mid x)$, produit de la vraisemblance et de la probabilité donnée par D. Pour prendre une décision à court terme (choix du prochain token x_t au temps t) qui est prometteuse à long terme, nous utilisons le Monte Carlo Tree Search (MCTS), déjà utilisé pour l'exploration d'espace de recherche pour les jeux (Silver *et al.*, 2018). En TAL, le MCTS a été très récemment utilisé pour la traduction automatique (Leblond *et al.*, 2021), la génération de questions et le résumé (Scialom *et al.*, 2021).

Une illustration d'un tel arbre peut être trouvée dans la Fig. 1, où la vraisemblance de x est calculée en multipliant les probabilités conditionnelles correspondantes le long du chemin, et la probabilité de classification est calculée au nœud terminal.

Le MCTS est un algorithme itératif dont chaque itération est constituée de quatre étapes consécutives. Dans le contexte particulier de l'application du MCTS à la génération de textes, nous avons procédé à quelques adaptations :

1 - Sélection Choisir récursivement les enfants depuis la racine jusqu'à un nœud qui n'a pas encore été développé. Pour n'explorer que les séquences viables, la probabilité $p_\theta(x_i \mid x_{1:t-1})$ d'un token x_i donné par le LM est utilisée pour cette phase. À cette fin, les enfants choisis sont ceux qui maximisent le Polynomial Upper Confidence Tree (PUCT) (Rosin, 2011; Silver *et al.*, 2017) :

$$PUCT(i) = \frac{s_i}{n_i} + c_{puct}\ p_\theta(x_i \mid x_{1:t-1}) \frac{\sqrt{N_i}}{1+n_i} \tag{1}$$

avec s_i le score agrégé du nœud i, n_i le nombre de simulations jouées après ce nœud, N_i le nombre de simulations jouées après son parent, et c_{puct} une constante définissant le compromis entre exploitation de bonnes séquences et exploration de séquences prometteuses. Dans notre cas, nous définissons le score d'une séquence comme la probabilité $p(x \mid c)$.

2 - Expansion Si le nœud sélectionné n'est pas terminal, utiliser le LM pour le développer en créant ses enfants.

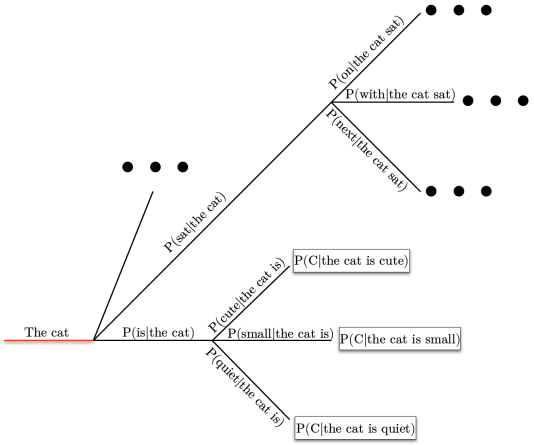

FIGURE 1 – Illustration du processus de génération sous contrainte sous la forme d'une exploration d'arbre à partir de l'amorce `The cat`. Les probabilités de classification ne sont représentées que sur les séquences terminées.

3 - Simulation (roll-out) Échantillonner l'un de ces enfants selon $p_\theta(x_i \mid x_{1:t-1})$, et aller à un nœud terminal (par une marche aléatoire ou autre).

4 - Rétro-propagation Agréger le score final obtenu au nœud terminal à chaque parent jusqu'à la racine. Parmi les différentes stratégies imaginables pour le faire, nous choisissons le score agrégé s_i associé au nœud i comme la probabilité moyenne sur toutes les simulations jouées après ce noeud.

Lorsque le nombre d'itérations a atteint le budget alloué, la construction de l'arbre s'arrête. Pour sélectionner le token x_i pour l'étape de décodage en cours nous choisissons le nœud le plus joué parmi les nœuds enfants de la racine.

Ces adaptations du MCTS à la génération sous contrainte sont résumées dans la Fig. 2. Notez que tout modèle de langue peut être utilisé pour définir $p_\theta(x_i \mid x_{1:t-1})$ et tout discriminateur pour évaluer les séquences, d'où le nom de notre approche : Plug and Play Language - Monte Carlo Tree Search (PPL-MCTS).

Afin de permettre un contrôle plus fin de la façon dont la contrainte est appliquée, nous introduisons un paramètre $\alpha \in [0, 1]$ pour contrôler le compromis entre la vraisemblance de la séquence et la force de la contrainte, en modifiant l'équation de Bayes : $p(x \mid c) \propto p_D(c \mid x)^\alpha p_\theta(x)^{1-\alpha}$. Notez que les PUCT (1) prennent déjà en compte la vraisemblance de la séquence, favorisant la sélection de nœuds à forte vraisemblance. Par conséquent, même les séquences générées avec $\alpha = 1$ sont correctement écrites. La définition de $\alpha < 1$ force l'algorithme à explorer des solutions encore plus proches du LM. Dans nos expériences, nous avons fixé $\alpha = 1$ pour renforcer la contrainte.

Pour éviter des roll-out coûteux, nous pouvons également attribuer une valeur aux séquences non terminées au prix d'une évaluation moins précise que ce qui émanerait d'un roll-out complet. En effet, le discriminateur peut être entraîné sur des séquences avec un nombre variable de tokens, ce qui permet de l'utiliser à chaque nœud sans avoir recours à des simulations. Dans cette configuration, le MCTS est utilisé comme un compromis efficace entre l'exploration et l'exploitation, perdant une partie de sa propriété de vision à long terme mais permettant d'orienter l'exploration vers des

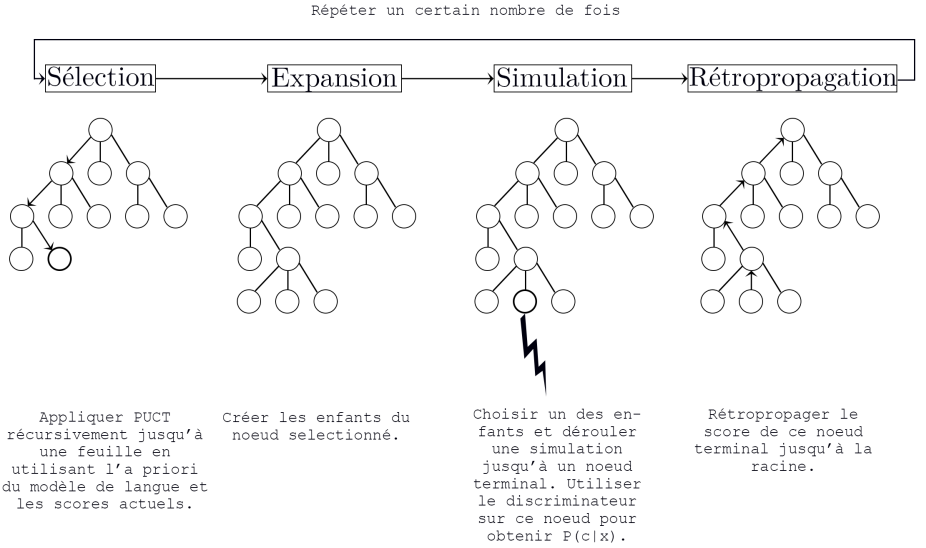

FIGURE 2 – Application du MCTS à la génération de texte.

solutions prometteuses.

4 Expérimentations

4.1 Tâches, jeux de données et mesures de performance

Trois jeux de données différents sont utilisés dans nos expériences : amazon_polarity (Zhang *et al.*, 2015), CLS (issu de FLUE (Le *et al.*, 2020)) et emotion (Saravia *et al.*, 2018). Les deux premiers sont des critiques Amazon (resp. en anglais et en français) étiquetées selon leur polarité positive ou négative ; la tâche est donc de générer avec la polarité voulue. Emotion est une collection de tweets classés en huit émotions : colère, anticipation, dégoût, peur, joie, tristesse, surprise et confiance. Ce jeu de données est supposé être plus difficile car il y a plus de classes et les textes sont plus petits (composés d'une seule phrase), donc le modèle doit générer précisément l'émotion cible avec peu de tokens. Il convient de noter que les trois jeux de données ont des tailles différentes : 4 000 000 d'instances au total pour amazon_polarity, 20 000 pour emotion et 6 000 pour CLS.

L'objectif de la génération sous contrainte est de générer des échantillons qui 1) appartiennent à une classe spécifique, tout en 2) conservant la qualité linguistique du LM original, et 3) avec une diversité suffisante entre les échantillons. Nous avons choisi des métriques connues pour chacun de ces aspects :

1. la précision, évaluée via un classifieur "oracle" externe (entraîné sur un ensemble de données disjoint de celui utilisé pour entraîner D) ;
2. la perplexité, calculée avec un LM "oracle" (LM non contraint entraîné sur des données différentes de celles du LM guidé) ;

3. le Self-BLEU (Zhu *et al.*, 2018), qui est le score BLEU (Papineni *et al.*, 2002) d'une séquence en utilisant les autres séquences comme références ; un Self-BLEU élevé signifie qu'il y a beaucoup de redondances entre les échantillons générés, et donc que la diversité est faible.

De telles métriques automatiques ont des limites connues (Caccia *et al.*, 2020) mais les résultats de l'évaluation humaine sur le jeu de données CLS, détaillés dans la section 4.6, confirment qu'elles fournissent un bon aperçu des performances. En pratique, le jeu de données étudié (voir ci-dessous) est divisé en deux parties, chaque partie étant subdivisée en ensembles entraînement/validation/test. La première partie sert à entraîner les modèles utilisés pour la génération (LM et discriminateur), tandis que la seconde est utilisée pour entraîner les oracles qui servent à calculer les métriques d'évaluation automatique. L'ensemble de test de la deuxième partie sera également utilisé pour forger des oracles pour la génération. Chaque métrique est évaluée sur un ensemble de 900 séquences générées, dont les amorces sont fonction de chaque jeu des données. Pour amazon_polarity, ce sont les titres des critiques qui servent d'amorce. Pour les deux autres jeux, les amorces sont les tout premiers tokens du champ texte. Les textes d'emotion et de CLS ayant des longueurs différentes, la taille de ces amorces est différente : elle est arbitrairement fixée à 6 tokens pour CLS et 4 pour emotion. Un exemple de génération à partir de amazon_polarity est donné en fig. 3.

<|startoftext|> The Revenge of making a good Halloween film. [SEP]?????? I think this movie is a waste of time. It's not scary, it's just plain stupid. The only good thing about this film is the soundtrack.<|endoftext|>
<|startoftext|> The Revenge of making a good Halloween film. [SEP] ive seen this movie a few times and i love it. the acting is great, the story line is good, and the special effects are awesome. if you like horror movies then go see this one.<|endoftext|>

FIGURE 3 – Exemple de deux générations contraintes utilisant PPL-MCTS, l'une sur la classe négative, l'autre sur la classe positive, en utilisant la même amorce (en gras) de amazon_polarity.

4.2 Méthodes

Méthodes simples. En plus de PPL-MCTS, nous proposons plusieurs techniques sur la base du réordonnancement, avec différentes variantes dans la façon dont 1) les séquences à réordonner sont produites et 2) la séquence finale est choisie. Trois méthodes sont testées pour générer des propositions : *beam search* (Dept., 2018) (avec un faisceau de taille 3), échantillonnage du noyau (*nucleus sampling, top-p sampling*) (Holtzman *et al.*, 2020) (avec p=0,9), ainsi que l'échantillonnage par faisceau (*beam sampling* (Caccia *et al.*, 2020)). Nous proposons aussi 3 méthodes pour le choix final : *argmax*, où la séquence qui a le $p(x|c)$ le plus élevé est choisie ; *first true*, où les propositions sont triées par vraisemblance décroissante et la première séquence qui, selon D, appartient à la bonne classe est choisie ; et *sampling*, où la distribution de $p(x|c)$ pour les propositions est normalisée et la séquence choisie est échantillonnée suivant cette distribution.

Méthodes de l'état-de-l'art. Nous comparons nos résultats avec les méthodes de la littérature. En particulier, nous testons les CC-LMs entraînés sur la tâche cible, de manière similaire à CTRL. Nous proposons également une implémentation de CC-LM entraînée avec la *classification loss* (notée *classloss*) initialement proposée pour la méthode GeDi (Krause *et al.*, 2020). Ces CC-LM sont également utilisés pour implémenter le modèle état-de-l'art GeDi. Nous rapportons les résultats des modèles GeDi entraînés avec et sans *classification loss*. Enfin, nous présentons les résultats du modèle PPLM. Pour une comparaison équitable, le même D et le même LM sont utilisés pour notre approche PPL-MCTS, nos approches par reclassement et PPLM.

4.3 Protocole expérimental

Pour chaque méthode, un nombre de tokens égal à la longueur moyenne des séquences du jeu de données est généré : 98 tokens pour amazon_polarity, 23 pour emotion et 137 pour CLS. Cela permet des comparaisons équitables entre les méthodes (notamment pour la perplexité et Self-BLEU).

Pour les expériences, trois modèles sont nécessaires : 1) un discriminateur (BERT-base-cased (Devlin *et al.*, 2019) pour l'anglais, FlauBERT-large-cased (Le *et al.*, 2020) pour le français), 2) un LM utilisé comme générateur (GPT-2 small pour l'anglais et belgpt2 pour le français, entraîné sur les jeux de données sans étiquetage), et 3) le CC-LM utilisé dans les approches CTRL et GeDi (version ré-entraînée du LM classique auquel est ajouté le code de contrôle).

Pour chaque méthode, nous avons testé plusieurs valeurs de température (1,0, 1,1 et 1,2), et de c_{puct} pour PPL-MCTS (1,0, 3,0, 5,0 et 8,0). Nous ne rapportons les résultats que pour les paramètres donnant les meilleures précisions. Le nombre d'itérations du MCTS par token et le nombre de propositions pour le réordonnancement sont fixés à 50, sauf pour le beam sampling où il est fixé à 10 en raison des limitations de mémoire. Le roll-out étant coûteux pour les longues séquences, nous ne l'appliquons que sur le jeu de données emotion afin de pouvoir mener des expériences approfondies. Sans roll-out, le MCTS perd une partie de sa propriété de vision à long terme mais permet toujours d'orienter l'exploration vers des solutions prometteuses. L'étude de l'impact de la longueur du roll-out est détaillée en section 4.7 Les autres paramètres des modèles de la littérature sont ceux fournis par les auteurs. Les expériences ont été menées sur une Quadro RTX 6000 avec 80 Go de RAM.

4.4 Résultats

Les résultats rapportés dans la Table 1 montrent que PPL-MCTS est compétitif pour la génération contrainte par rapport aux LM entraînés spécifiquement à la tâche sur l'aspect de l'application des contraintes (précision élevée), tout en conservant une grande diversité (faible Self-BLEU) et en restant proche de la distribution originale (faible perplexité). Sur les trois jeux de données et métriques, il donne constamment les meilleurs résultats ; la seule autre méthode très performante est GeDi avec *classification loss*. D'autres compromis précision/perplexité sont réalisables en modifiant le paramètre $\alpha \in [0, 1]$ pour contrôler l'équilibre entre la vraisemblance de la séquence et la force de la contrainte (voir section 4.8).

Un autre résultat remarquable est celui de la méthode Sampling - Argmax. Grâce à l'échantillonnage utilisé pour générer les propositions, son Self-BLEU est parmi les plus bas de toutes les méthodes. Malgré sa simplicité et son faible coût de calcul, sa précision est parmi les meilleures sur tous les jeux de données. La grande perplexité indique cependant que les échantillons générés peuvent être très différents de ceux générés par un LM standard sur ce jeu de données.

4.5 Exemples de générations

Nous fournissons des exemples de générations pour les jeux de données amazon_polarity et emotion obtenus avec les méthodes PPL-MCTS, PPLM, GeDi et Sampling - Argmax, respectivement dans les figures 4 et 5. Les textes générés par Sampling - Argmax sont assez différents, ce qui est confirmé par les hautes valeurs de perplexité notées précédemment. Notons que les textes pour emotion sont composés d'une seule phrase, alors que ceux d'amazon_polarity sont des revues de produits complètes.

Méthode de génération	amazon_polarity Précision oracle ↑	amazon_polarity 5 - Self-BLEU ↓	amazon_polarity Perplexité oracle ↓	emotion Précision oracle ↑	emotion 5 - Self-BLEU ↓	emotion Perplexité oracle ↓	CLS Précision oracle ↑	CLS 5 - Self-BLEU ↓	CLS Perplexité oracle ↓
LM ré-entraînés									
CC-LM - Classloss	0,82	0,79	**2,56***,†	**0,89***	0,65†	3,72*,†	0,89*	**0,04***,†	50,6
CC-LM	0,91	0,71	3,21†	0,52	**0,13***,†	11,1	0,66	0,06*,†	31,5
GeDi - Classloss	0,96*	0,6*	5,16	0,88*	0,68	5,57*	0,94*	0,4	7,99*
GeDi	0,96*	0,6*	5,16	0,54	0,52†	4,09*,†	0,83*	0,31†	11,9
LM sans ré-entraînement									
PPLM	0,89	0,66	2,84†	0,67	0,19†	7,31	0,79	0,23†	8,3
Beam - Argmax	0,88	0,85	3,14†	0,72*	0,49†	3,7*,†	0,64	0,82	3,31*,†
Beam - Sampling	0,86	0,84	3,27†	0,7	0,46†	3,69*,†	0,6	0,82	3,37*,†
Beam - First true	0,85	0,83	3,27†	0,65	0,38†	**3,68***,†	0,62	0,82	**3,26***,†
Beam sampling - Argmax	0,97*	0,73	3,82†	0,67	0,48†	3,88*,†	0,88*	0,67	3,91*,†
Beam sampling - Sampling	0,92	0,76	3,68†	0,66	0,48†	3,88*,†	0,76	0,63	4,07*,†
Beam sampling - First true	0,9	0,73	3,84†	0,66	0,49†	3,85*,†	0,85*	0,71	3,8*,†
Sampling - Argmax	**0,99***,†	0,17*,†	16,5	0,87*	**0,13***,†	11,7	0,92*	0,12*,†	14,3
Sampling - First true	0,89	**0,07***,†	85,9	0,82*	**0,13***,†	10,4	0,87*	0,14*,†	13
Sampling - Sampling	0,88	0,17*,†	16,3	0,81*	**0,13***,†	10,4	0,81	0,06*,†	31,8
PPL-MCTS	0,97*	0,63*	5,69	0,84*	0,37†	4,82*,†	0,89*	0,54	4,98*,†
PPL-MCTS - 10 tokens roll-out	–	–	–	–	–	–	**0,95***	0,57	5,07*,†

TABLE 1 – Performances des méthodes de génération sous contrainte sur amazon_polarity, emotion et CLS. Le symbole † (resp. $*$) indique une amélioration statistiquement significative (p=0,01) par rapport à GeDi-classloss (resp. PPLM).

Cela permet d'apprécier les capacités des modèles dans des contextes différents.

PPL-MCTS
<|startoftext|> realistic account [SEP] ive read this book and i think it is a very good account of the life of a young man. i would recommend to anyone who likes to know about the lives of people in america, or any other country that has been around for centuries.<|endoftext|>

PPLM
<|startoftext|> realistic account [SEP] I enjoyed this book. It was realistic and I enjoyed the way the author described the people and places. I would recommend this book to anyone who is interested in the history of the United States.<|endoftext|>

GeDi
<|startoftext|> realistic account [SEP] This is a real book. It's not just about the war, but also about the people who fought it. The author has a very realistic view of what happened in Vietnam and how it affected our country. I would recommend this book to anyone interested in learning more about Vietnam or any other war that took place during that time period<|endoftext|>

Sampling - Argmax
<|startoftext|> realistic account [SEP] ive seen many of these. they mostly in the 20's, 30's and 40's and the luscious ones i know - men like this - there's an old saying that farts, u makin' u sell it..this movie has a lot of realism to it too ! and i was totally impressed on how good the kids and the predator was ! will it be hard for them to make more like this ? i think it will ! i read that war is going to be much<|endoftext|>

FIGURE 4 – Exemples de génération contrainte avec les méthodes PPL-MCTS, PPLM, GeDi et Sampling - Argmax (de haut en bas) sur la classe 'positive' du jeu de données amazon_polarity ; le prompt (en gras) est le même à chaque fois.

4.6 Évaluation humaine

Nous proposons une évaluation humaine pour comparer avec les résultats obtenus par les oracles et confirmer les résultats et la pertinence des métriques automatiques. En raison du coût de l'annotation, nous limitons les méthodes testées aux deux méthodes état-de-l'art (PPLM et GeDi), à PPL-MCTS et à la prometteuse Sampling - Argmax. Cela permet de tester si PPL-MCTS est effectivement aussi efficace que GeDi et si les deux sont meilleures que PPLM. En outre, cela devrait confirmer que la perplexité élevée de la méthode Sampling - Argmax est due au fait que les textes générés sont très différents de ceux générés par les autres méthodes. L'évaluation a été réalisée sur le jeu de données

PPL-MCTS
<|startoftext|> i feel that working with a group of people who are so passionate about the same thing is really important<|endoftext|>

PPLM
<|startoftext|> i feel that working hard and caring for someone i don t care for is a lot less selfish than i would be feeling for someone i<|endoftext|>

GeDi
<|startoftext|> i feel that working with the ladies of the family is a wonderful thing and i am very fond of the way they look and feel<|endoftext|>

Sampling - Argmax
<|startoftext|> i feel that working at imgur for so many years is ill be devoted to it<|endoftext|>

FIGURE 5 – Exemples de génération contrainte avec les méthodes PPL-MCTS, PPLM, GeDi et Sampling - Argmax (de haut en bas) sur la classe 'love' du jeu de données emotion ; le prompt (en gras) est le même à chaque fois.

CLS par trois collègues volontaires, de langue maternelle française. Ils ont étiqueté le même ensemble de critiques afin de mesurer l'accord inter-annotateurs. Les annotateurs étant de langue maternelle française, l'évaluation a été réalisée sur le jeu de données CLS.

L'échantillon est composé de 50 commentaires (25 positifs et 25 négatifs) choisis au hasard pour chaque méthode, ce qui donne un total de 200 commentaires. On a demandé aux annotateurs de passer en revue cet ensemble (mélangé au hasard) et de donner deux notes (entre 1 et 5) pour chaque critique :

1. **Polarité**. Le texte est noté 5 s'il correspond entièrement à la polarité attendue, jusqu'à 1 s'il correspond entièrement à l'étiquette opposée. Ce score correspond à la précision dans les métriques automatiques.
2. **Lisibilité**. Le texte est noté 5 s'il ne contient aucune erreur (grammaire, logique...) et parfaitement compréhensible. Plus il y a d'erreurs ou d'incohérences, plus le score est faible. Ce score correspond à la perplexité dans les métriques automatiques.

La diversité au sein de l'ensemble de textes générés est compliquée à évaluer manuellement et Self-BLEU est assez précis en tant que mesure de diversité, cette propriété n'est donc pas étudiée dans l'évaluation humaine.

Nous rapportons les scores moyens des 3 annotateurs ainsi que l'écart-type dans le Tableau 2. Un t-test contre PPLM (GeDi étant le meilleur sur les deux scores) est utilisé pour évaluer la significativité statistique (p-valeur à 1 %). On peut remarquer que l'accord entre les annotateurs est élevé et que les résultats sont en accord avec les conclusions des métriques automatiques. GeDi, lorsqu'il est entraîné avec la perte de classification, donne des résultats similaires à ceux de PPL-MCTS, en termes de

Génération	Polarité	Lisibilité
GeDi - Classloss	$4,46 \pm 0,08^*$	$4,19 \pm 0,28^*$
PPL-MCTS	$4,43 \pm 0,12^*$	$4,05 \pm 0,23^*$
PPLM	$3,74 \pm 0,08$	$3,12 \pm 0,19$
Sampling - Argmax	$4,00 \pm 0,11$	$2,83 \pm 0,33$

TABLE 2 – Résultats de l'évaluation humaine sur le jeu de données CLS (moyenne sur 3 annotateurs). $*$ indique une amélioration statistiquement significative ($p \leq 1\%$) par rapport à PPL M.

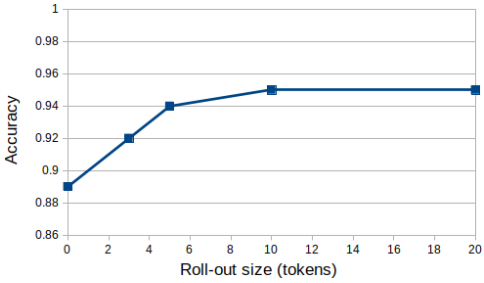

FIGURE 6 – Précision en fonction de la taille du roll-out ; jeu de données CLS

satisfaction des contraintes et de qualité d'écriture. PPLM, en revanche, génère des échantillons de moindre qualité et a plus de difficultés à appliquer la contrainte. Enfin, compte tenu de son score de lisibilité, Sampling - Argmax semble générer des échantillons de faible qualité. Son score de polarité, bien qu'il soit supérieur à celui de PPLM, est plus faible que prévu : étant donné la précision rapportée par l'oracle, il devrait être proche de GeDi et de PPL-MCTS. Cela est très probablement dû au fait qu'il est difficile pour un humain d'évaluer la polarité d'un texte mal écrit, ce qui entraîne souvent une évaluation neutre.

4.7 Effet du roll-out

Le roll out est coûteux pour les très longues séquences, et la question de son utilité se pose nécessairement. Nous étudions son influence pour un nombre fixe de tokens (au lieu d'aller jusqu'à la fin de la séquence) sur les performances de PPL-MCTS. Pour cette expérience, nous utilisons le jeu de données CLS et testons un roll-out à 0 (résultat original), 3, 5, 10 et 20 tokens. Comme on peut le constater sur la Fig. 6, seulement 5 tokens permettent à PPL-MCTS d'être à égalité avec GeDi sur ce jeu de données. La taille du roll-out améliore rapidement la précision, qui atteint ensuite un plateau. Cela suggère que le fait d'avoir un horizon est vraiment utile, mais seulement jusqu'à un certain point. De plus, les valeurs de perplexité et Self-BLEU restent stables, variant respectivement de 0,54 à 0,57, et de 4,98 à 5,18 lorsque la taille du roll-out augmente de 0 à 20 % (non montré sur la Fig. 6). La taille du roll-out peut donc être fixée en fonction du budget de calcul, ce qui permet de mieux définir le compromis entre coût et qualité.

4.8 Force de la contrainte avec le paramètre α

Comme expliqué précédemment, un paramètre α permet de contrôler l'importance relative donnée au score du discriminateur vis-à-vis de la vraisemblance du LM. Notons que dans les expériences précédentes, on avait fixé $\alpha = 1$ pour mettre l'accent sur le respect de la contrainte, sachant que les méthodes de décodage proposées permettaient par essence de générer des textes avec une bonne vraisemblance.

Ici, nous testons sur amazon_polarity d'autres valeurs pour α avec la méthode Sampling - Argmax. Nous choisissons cette méthode sur ce jeu de données puisqu'elle obtient la meilleure précision mais a une forte perplexité. Dans ce cas, il semble intéressant de rechercher un compromis en sacrifiant un peu de précision pour des textes mieux écrits.

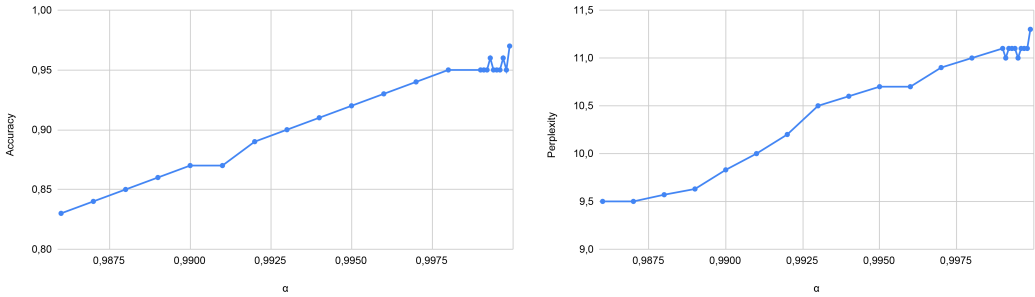

FIGURE 7 – Précision (gauche) et perplexité (droite) de Sampling - Argmax selon le paramètre α ; données amazon_polarity.

Les résultats sont stables avec $\alpha < 0,98$, l'impact se fait sentir uniquement pour des valeurs entre 0,98 et 1. Cela s'explique par le fait que, pour une séquence suffisamment longue, $p_\theta(x) \ll p_D(c \mid x)$, rendant l'effet de α négligeable. Comme visible dans la figure 7, pour les valeurs entre 0,98 et 1, on peut en revanche observer une dépendance linéaire de α avec la précision et avec la perplexité. Cela illustre bien la pertinence de ce paramètre pour définir la respect souhaité de la contrainte, et, indirectement, l'éloignement à la distribution du modèle de langue original.

5 Conclusion

Contrôler la génération à l'aide d'un discriminateur est flexible et très utile lors de l'utilisation de très grands modèles de langage, tels que GPT-3, dont les coûts de calcul pour le *fine-tuning* sont prohibitifs. En revanche, l'apprentissage d'un discriminateur est plus facile et moins coûteux. Les méthodes de contrôle de génération que nous proposons, ajoutant la contrainte du discriminateur durant la génération, donnent des performances équivalentes aux meilleures approches basées sur le ré-entraînement de LM. Le MCTS permet de définir précisément le meilleur compromis entre coût et qualité à travers le nombre d'itérations et la taille du roll-out, tout en assurant théoriquement l'efficacité de la recherche. Dans un souci de reproductibilité, notre implémentation est mise à disposition sur `https://github.com/NohTow/PPL-MCTS`.

En considérant la génération de texte comme un processus d'exploration d'arbre, l'approche GeDi réduit le coût de l'exploration en largeur mais l'exploration en profondeur reste un problème. Son utilisation pour la génération sous contrainte reste donc similaire à une recherche standard du maximum de vraisemblance, pour lequel on ne dispose pas d'une méthode optimale. À l'inverse, le MCTS offre un moyen efficace d'explorer l'arbre de recherche en déterminant le meilleur choix local à long terme, ce qui réduit le coût de l'exploration en profondeur. Ainsi, ces deux méthodes résolvent différentes facettes de la génération sous contrainte, et la combinaison des deux est une perspective prometteuse.

Plusieurs pistes de recherche sont ouvertes par ce travail. Ainsi, une taille de roll-out adaptative, par exemple jusqu'à ce que le score du discriminateur soit supérieur ou inférieur à un seuil comme dans (Cotarelo *et al.*, 2021), semblerait particulièrement adaptée aux textes.

Références

BROWN T. B., MANN B., RYDER N., SUBBIAH M., KAPLAN J., DHARIWAL P., NEELAKANTAN A., SHYAM P., SASTRY G., ASKELL A., AGARWAL S., HERBERT-VOSS A., KRUEGER G., HENIGHAN T., CHILD R., RAMESH A., ZIEGLER D. M., WU J., WINTER C., HESSE C., CHEN M., SIGLER E., LITWIN M., GRAY S., CHESS B., CLARK J., BERNER C., MCCANDLISH S., RADFORD A., SUTSKEVER I. & AMODEI D. (2020). Language models are few-shot learners. In H. LAROCHELLE, M. RANZATO, R. HADSELL, M. BALCAN & H. LIN, Éds., *Advances in Neural Information Processing Systems 33 : Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

CACCIA M., CACCIA L., FEDUS W., LAROCHELLE H., PINEAU J. & CHARLIN L. (2020). Language gans falling short. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* : OpenReview.net.

COTARELO A., GARCÍA DÍAZ V., NÚÑEZ VALDEZ E., GONZÁLEZ GARCÍA C., GÓMEZ A. & LIN J. (2021). Improving monte carlo tree search with artificial neural networks without heuristics. *Applied Sciences*, **11**, 2056. DOI : 10.3390/app11052056.

COULOM R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In H. J. VAN DEN HERIK, P. CIANCARINI & H. H. L. M. DONKERS, Éds., *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, volume 4630 de *Lecture Notes in Computer Science*, p. 72–83 : Springer. DOI : 10.1007/978-3-540-75538-8_7.

DATHATHRI S., MADOTTO A., LAN J., HUNG J., FRANK E., MOLINO P., YOSINSKI J. & LIU R. (2020). Plug and play language models : A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* : OpenReview.net.

DEPT. C.-M. U. S. (2018). Speech understanding systems : summary of results of the five-year research effort at carnegie-mellon university. DOI : 10.1184/R1/6609821.v1.

DEVLIN J., CHANG M., LEE K. & TOUTANOVA K. (2019). BERT : pre-training of deep bidirectional transformers for language understanding. In J. BURSTEIN, C. DORAN & T. SOLORIO, Éds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, p. 4171–4186 : Association for Computational Linguistics. DOI : 10.18653/v1/n19-1423.

HOLTZMAN A., BUYS J., DU L., FORBES M. & CHOI Y. (2020). The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020* : OpenReview.net.

HOLTZMAN A., BUYS J., FORBES M., BOSSELUT A., GOLUB D. & CHOI Y. (2018). Learning to write with cooperative discriminators. In I. GUREVYCH & Y. MIYAO, Éds., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1 : Long Papers*, p. 1638–1649 : Association for Computational Linguistics. DOI : 10.18653/v1/P18-1152.

KESKAR N. S., MCCANN B., VARSHNEY L. R., XIONG C. & SOCHER R. (2019). CTRL : A conditional transformer language model for controllable generation. *CoRR*, **abs/1909.05858**.

KRAUSE B., GOTMARE A. D., MCCANN B., KESKAR N. S., JOTY S. R., SOCHER R. & RAJANI N. F. (2020). Gedi : Generative discriminator guided sequence generation. *CoRR*, **abs/2009.06367**.

KUMAR V., CHOUDHARY A. & CHO E. (2020). Data augmentation using pre-trained transformer models. *CoRR*, **abs/2003.02245**.

LE H., VIAL L., FREJ J., SEGONNE V., COAVOUX M., LECOUTEUX B., ALLAUZEN A., CRABBÉ B., BESACIER L. & SCHWAB D. (2020). Flaubert : Unsupervised language model pre-training for french. In N. CALZOLARI, F. BÉCHET, P. BLACHE, K. CHOUKRI, C. CIERI, T. DECLERCK, S. GOGGI, H. ISAHARA, B. MAEGAARD, J. MARIANI, H. MAZO, A. MORENO, J. ODIJK & S. PIPERIDIS, Éds., *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, p. 2479–2490 : European Language Resources Association.

LEBLOND R., ALAYRAC J., SIFRE L., PISLAR M., LESPIAU J., ANTONOGLOU I., SIMONYAN K. & VINYALS O. (2021). Machine translation decoding beyond beam search. In M. MOENS, X. HUANG, L. SPECIA & S. W. YIH, Éds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, p. 8410–8434 : Association for Computational Linguistics.

PAPANIKOLAOU Y. & PIERLEONI A. (2020). DARE : data augmented relation extraction with GPT-2. *CoRR*, **abs/2004.13845**.

PAPINENI K., ROUKOS S., WARD T. & ZHU W. (2002). Bleu : a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, p. 311–318 : ACL. DOI : 10.3115/1073083.1073135.

RADFORD A., WU J., CHILD R., LUAN D., AMODEI D. & SUTSKEVER I. (2019). Language models are unsupervised multitask learners.

ROSIN C. D. (2011). Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.*, **61**(3), 203–230. DOI : 10.1007/s10472-011-9258-6.

SARAVIA E., LIU H. T., HUANG Y., WU J. & CHEN Y. (2018). CARER : contextualized affect representations for emotion recognition. In E. RILOFF, D. CHIANG, J. HOCKENMAIER & J. TSUJII, Éds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, p. 3687–3697 : Association for Computational Linguistics. DOI : 10.18653/v1/d18-1404.

SCIALOM T., DRAY P., LAMPRIER S., PIWOWARSKI B. & STAIANO J. (2020). Discriminative adversarial search for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 de *Proceedings of Machine Learning Research*, p. 8555–8564 : PMLR.

SCIALOM T., DRAY P., LAMPRIER S., PIWOWARSKI B. & STAIANO J. (2021). To beam or not to beam : That is a question of cooperation for language gans. *Advances in neural information processing systems*.

SILVER D., HUBERT T., SCHRITTWIESER J., ANTONOGLOU I., LAI M., GUEZ A., LANCTOT M., SIFRE L., KUMARAN D., GRAEPEL T., LILLICRAP T., SIMONYAN K. & HASSABIS D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, **362**(6419), 1140–1144. DOI : 10.1126/science.aar6404.

SILVER D., SCHRITTWIESER J., SIMONYAN K., ANTONOGLOU I., HUANG A., GUEZ A., HUBERT T., BAKER L., LAI M., BOLTON A., CHEN Y., LILLICRAP T. P., HUI F., SIFRE L., VAN DEN DRIESSCHE G., GRAEPEL T. & HASSABIS D. (2017). Mastering the game of go without human knowledge. *Nat.*, **550**(7676), 354–359. DOI : 10.1038/nature24270.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L. & POLOSUKHIN I. (2017). Attention is all you need. In I. GUYON, U. VON LUXBURG, S. BENGIO, H. M. WALLACH, R. FERGUS, S. V. N. VISHWANATHAN & R. GARNETT, Éds., *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, p. 5998–6008.

WOLF T., DEBUT L., SANH V., CHAUMOND J., DELANGUE C., MOI A., CISTAC P., RAULT T., LOUF R., FUNTOWICZ M., DAVISON J., SHLEIFER S., VON PLATEN P., MA C., JERNITE Y., PLU J., XU C., SCAO T. L., GUGGER S., DRAME M., LHOEST Q. & RUSH A. M. (2020). Transformers : State-of-the-art natural language processing. In Q. LIU & D. SCHLANGEN, Éds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing : System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, p. 38–45 : Association for Computational Linguistics. DOI : 10.18653/v1/2020.emnlp-demos.6.

ZELLERS R., HOLTZMAN A., RASHKIN H., BISK Y., FARHADI A., ROESNER F. & CHOI Y. (2019). Defending against neural fake news. In H. M. WALLACH, H. LAROCHELLE, A. BEYGELZIMER, F. D'ALCHÉ-BUC, E. B. FOX & R. GARNETT, Éds., *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, p. 9051–9062.

ZHANG X., ZHAO J. J. & LECUN Y. (2015). Character-level convolutional networks for text classification. In C. CORTES, N. D. LAWRENCE, D. D. LEE, M. SUGIYAMA & R. GARNETT, Éds., *Advances in Neural Information Processing Systems 28 : Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, p. 649–657.

ZHU Y., LU S., ZHENG L., GUO J., ZHANG W., WANG J. & YU Y. (2018). Texygen : A benchmarking platform for text generation models. In K. COLLINS-THOMPSON, Q. MEI, B. D. DAVISON, Y. LIU & E. YILMAZ, Éds., *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, p. 1097–1100 : ACM. DOI : 10.1145/3209978.3210080.