La génération automatique de poésie en français

Tim van de Cruys

La génération automatique de poésie en français

Tim Van de Cruys
IRIT & CNRS
118 Route de Narbonne
31062 Toulouse Cedex 9
France
tim.vandecruys@irit.fr

RÉSUMÉ

La génération automatique de poésie est une tâche ardue pour un système informatique. Pour qu'un poème ait du sens, il est important de prendre en compte à la fois des aspects linguistiques et littéraires. Ces dernières années, un certain nombre d'approches fructueuses sont apparues, capables de modéliser de manière adéquate divers aspects du langage naturel. En particulier, les modèles de langue basés sur les réseaux de neurones ont amélioré l'état de l'art par rapport à la modélisation prédictive de langage, tandis que les *topic models* sont capables de capturer une certaine cohérence thématique. Dans cet article, on explorera comment ces approches peuvent être adaptées et combinées afin de modéliser les aspects linguistiques et littéraires nécessaires pour la génération de poésie. Le système est exclusivement entraîné sur des textes génériques, et sa sortie est contrainte afin de conférer un caractère poétique au vers généré. Le cadre présenté est appliqué à la génération de poèmes en français, et évalué à l'aide d'une évaluation humaine.

ABSTRACT

Automatic Poetry Generation in French

Automatic poetry generation is a challenging task for a computational system. For a poem to be meaningful, both linguistic and literary aspects need to be taken into account. In the last few years, a number of successful approaches have emerged that are able to adequately model various aspects of natural language. Particularly, language models based on neural networks have improved the state of the art with regard to predictive language modeling, while topic models are able to capture some form of thematic coherence. In this article, we will explore how these approaches can be adapted and combined to model the linguistic and literary aspects needed for poetry generation. The system is exclusively trained on generic text, and its output is constrained in order to confer a poetic character to the generated verse. The framework is applied to the generation of poems in French, and it is evaluated using a human evaluation.

MOTS-CLÉS : génération de poésie, réseaux de neurones, factorisation en matrices non-négatives.

KEYWORDS: poetry generation, neural networks, non-negative matrix factorization.

1 Introduction

La génération automatique de poésie est une tâche ardue pour un système informatique. Pour qu'un poème ait du sens, il est important de prendre en compte à la fois des aspects linguistiques et littéraires. Tout d'abord, un système de génération de poésie doit modéliser de manière correcte les phénomènes de langage, tels que la syntaxe, et la cohérence sémantique et discursive. De plus, le système doit intégrer diverses contraintes (telles que la forme et la rime) liées à un genre poétique particulier. Enfin, le système doit faire preuve d'une certaine créativité littéraire, ce qui rend le poème intéressant et digne d'être lu.

Ces dernières années, dans le domaine du traitement automatique des langues, un certain nombre d'approches fructueuses sont apparues, capables de modéliser de manière adéquate divers aspects du langage naturel. En particulier, les modèles de langue basés sur les réseaux de neurones ont amélioré l'état de l'art par rapport à la modélisation prédictive de langage, tandis que les *topic models* sont capables de capturer une certaine forme de cohérence thématique. Dans cet article, on explorera comment ces approches peuvent être adaptées et combinées afin de modéliser les aspects linguistiques et littéraires nécessaires pour la génération de poésie. Plus spécifiquement, dans ce travail, on utilisera des réseaux de neurones récurrents dans une configuration encodeur-décodeur. L'encodeur construit d'abord une représentation d'une phrase entière en incorporant séquentiellement les mots de cette phrase dans un vecteur d'état caché de taille fixe. La représentation finale est ensuite donnée au décodeur, qui émet une séquence de mots selon une distribution de probabilité dérivée de l'état caché de la phrase en entrée. En apprenant au réseau à prédire la phrase suivante avec la phrase actuelle en entrée, le réseau apprend à générer du texte brut avec une certain cohérence discursive. En transformant la distribution de probabilité fournie par le décodeur, afin d'incorporer des contraintes poétiques, le réseau peut être exploité pour la génération de vers poétiques. Il est important de noter que le système de poésie n'est pas entraîné sur des textes poétiques ; au contraire, le système est entraîné sur des textes génériques extraits du web, et ce seront alors les contraintes appliquées qui confèrent un caractère poétique aux vers générés.

Cet article est structuré comme suit. Dans la section 2, on présente un aperçu des travaux connexes sur la génération automatique de poésie. La section 3 décrit ensuite les différentes composantes du système de génération de poésie. Dans la section 4, on présentera un certain nombre d'exemples et une évaluation humaine. La section 5 conclut et examine quelques pistes pour des futurs travaux.

2 Travaux connexes

Il y a une longue et captivante histoire en termes de génération automatique de poésie pour le français (Queneau, 1961; OULIPO, 1981), que l'on qualifierait de créativité mécanique. Au-delà de la simple créativité mécanique, les premières implémentations informatiques se sont souvent appuyées sur des méthodes basées sur des règles ou sur des patrons. L'un des premiers exemples est le système ASPERA (Gervás, 2001) pour l'espagnol, qui repose sur une base de connaissances complexe, un ensemble de règles et un raisonnement à partir de cas. D'autres approches incluent Manurung *et al.* (2012), qui combinent la génération basée sur des règles avec des algorithmes génétiques ; le système de génération PoeTryMe de Gonçalo Oliveira (2012), qui repose sur la génération tabulaire (*chart generation*) et diverses stratégies d'optimisation ; et Veale (2013), qui exploite les expressions métaphoriques en utilisant une approche basée sur les patrons.

Alors que la génération de poésie avec des modèles basés sur des règles et des patrons a une tendance inhérente à être structurellement plutôt rigide, les progrès des méthodes statistiques pour la génération de langage ont ouvert de nouvelles perspectives pour une approche plus variée et hétérogène. Greene *et al.* (2010), par exemple, utilisent un modèle de langage n-gramme en combinaison avec un modèle rythmique implémenté avec des transducteurs à états finis. Et plus récemment, des réseaux de neurones récurrents ont été exploités pour la génération de la poésie. Zhang & Lapata (2014) utilise un encodeur-décodeur RNN pour la génération de poésie chinoise, dans lequel un premier RNN construit une représentation cachée du vers actuel dans un poème, et un deuxième RNN prédit le vers suivant mot par mot, en fonction de la représentation cachée du vers actuel. Le système est entraîné sur un corpus de poèmes chinois. Yan (2016) présente une amélioration de l'approche encodeur-décodeur en incorporant une méthode de raffinement itératif : le réseau construit un poème candidat à chaque itération, et la représentation de l'itération précédente est utilisée lors de la création de la suivante. Et Wang *et al.* (2016) étendent la méthode en utilisant un mécanisme d'attention.

Ghazvininejad *et al.* (2016) combinent des RNNs (afin de modéliser la fluidité syntaxique) avec des calculs de similarité distributionnelle (afin de modéliser la cohérence sémantique) et des automates à états finis (pour imposer des contraintes littéraires telles que le mètre et la rime). Leur système, HAFEZ, est capable de produire des poèmes bien formés avec un raisonnable degré de cohérence sémantique, basés sur un sujet défini par l'utilisateur. Hopkins & Kiela (2017) se concentrent sur les vers rythmiques ; ils combinent un RNN, entraîné sur une représentation phonétique de poèmes, avec une cascade de transducteurs à états finis pondérés. Et Lau *et al.* (2018) présentent un modèle de réseaux de neurones pour la génération de sonnets, qui intègre l'entraînement de la rime et du rythme dans le réseau ; le réseau apprend les motifs de stress iambiques à partir de données, tandis que les paires de mots qui riment sont séparées des paires de mots qui ne riment pas en utilisant une perte basée sur la marge.

Il est à noter que tous les modèles statistiques existants sont entraînés sur un corpus de poésie ; à notre connaissance, notre système est le premier à ne réaliser la génération de poésie qu'avec un modèle exclusivement entraîné sur un corpus générique, ce qui signifie que le caractère poétique est conféré par le modèle lui-même. Deuxièmement, on utilise un modèle sémantique latent pour modéliser la cohérence thématique, ce qui est également nouveau.

3 Modèle

3.1 Architecture neuronale

À la base du système de poésie se trouve un modèle de langage neuronal, entrainé à prédire la phrase suivante S_{i+1} à partir de la phrase courante S_i. L'architecture neuronale est composée de deux réseaux de neurones récurrents à portes (*gated recurrent units*, ou *GRUs* ; Cho *et al.*, 2014) fonctionnant dans une configuration encodeur-décodeur. L'encodeur prend en séquence chaque mot $w^i_{1,\ldots,N}$ de la phrase courante S_i (représenté par son plongement de mot ou *word embedding* $\mathbf{x}$) de manière qu'à chaque pas de temps t_i un état caché $\mathbf{h}_t$ est créé à la base du plongement du mot courant $\mathbf{x}_t$ et l'état caché $\mathbf{h}_{t-1}$ du pas de temps précédent. Pour chaque pas de temps, l'état caché $\hat{\mathbf{h}}_t$ est calculé selon les équations suivantes :

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \hat{\mathbf{h}}_{t-1}) \quad (1)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \hat{\mathbf{h}}_{t-1}) \quad (2)$$

$$\bar{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \hat{\mathbf{h}}_{t-1})) \quad (3)$$

$$\hat{\mathbf{h}}_t = (1 - \mathbf{z}_t) \odot \hat{\mathbf{h}}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t \quad (4)$$

où $\mathbf{r}_t$ représente la porte de réinitialisation du GRU, $\mathbf{z}_t$ représente la porte de mise à jour, $\bar{\mathbf{h}}_t$ représente le nouveau état candidat, et $\odot$ représente la multiplication élément par élément.

$\mathbf{h}_t$ peut être interprété comme une représentation de la séquence $w_1, \ldots, w_t$, et l'état caché final $\mathbf{h}_N$ sera donc une représentation de la phrase entière. Cet état caché final est ensuite donné comme entrée au décodeur. Le décodeur fait alors une prédiction mot par mot de la phrase suivante, conditionnée sur l'encodeur ; à chaque pas de temps t_{i+1}, le décodeur crée également un état caché $\mathbf{h}_t$ à la base du plongement $\mathbf{x}_t$ du mot courant (prédit par le décodeur dans le pas précédent) et l'état caché $\mathbf{h}_{t-1}$ du pas de temps précédent (le premier état caché étant $\mathbf{h}_N$ qui vient de l'encodeur et le premier mot étant un symbole d'initialisation). Les calculs pour chaque pas de temps $\mathbf{h}_t$ du décodeur sont égaux à ceux utilisés dans l'encodeur (équations 1 à 4).

Afin d'exploiter pleinement la séquence complète de représentations fournie par l'encodeur, l'architecture de base est complétée par un mécanisme d'attention, notamment l'attention dite *générale* (Luong *et al.*, 2015). Le mécanisme d'attention permet au décodeur de consulter l'ensemble des états cachés calculés par l'encodeur ; à chaque pas de temps – pour la génération de chaque mot de la phrase S_{i+1} – le décodeur détermine quels mots de la phrase S_i sont pertinents et sélectionne en conséquence une combinaison linéaire de l'ensemble des états cachés. À cette fin, on calcule d'abord un vecteur d'attention $\mathbf{a}_t$, qui attribue un poids à chaque état masqué $\hat{\mathbf{h}}_i$ produit par l'encodeur (en fonction de l'état caché actuel du décodeur $\mathbf{h}_t$) selon l'équation 5 :

$$\mathbf{a}_t(i) = \frac{\exp(\text{score}(\mathbf{h}_t, \hat{\mathbf{h}}_i))}{\sum_{i'} \exp(\text{score}(\mathbf{h}_t, \hat{\mathbf{h}}_{i'}))} \quad (5)$$

où

$$\text{score}(\mathbf{h}_t, \hat{\mathbf{h}}_i) = \mathbf{h}_t^T \mathbf{W}_\mathbf{a} \hat{\mathbf{h}}_i \quad (6)$$

L'étape suivante consiste à calculer un vecteur de contexte global $\mathbf{c}_t$, qui est une moyenne pondérée (basée sur le vecteur d'attention $\mathbf{a}_t$) de tous les états masqués de l'encodeur. Le vecteur de contexte qui en résulte est ensuite combiné avec l'état caché du décodeur d'origine afin de calculer un nouvel état caché augmenté avec l'attention, $\tilde{\mathbf{h}}_t$:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_\mathbf{c}[\mathbf{c}_t; \mathbf{h}_t]) \quad (7)$$

où $[\cdot;\cdot]$ représente la concaténation des vecteurs. Enfin, l'état caché qui en résulte $\tilde{\mathbf{h}}_t$ est transformé en distribution de probabilité $p(\mathbf{w}^t | w^{<t}, S_i)$ sur le vocabulaire entier en utilisant une couche softmax.

$$p(\mathbf{w}^t | w^{<t}, S_i) = \text{softmax}(\mathbf{W}_\mathbf{s} \tilde{\mathbf{h}}_t) \quad (8)$$

Comme fonction objective, on optimise la somme des log-probabilités de la phrase suivante, conditionnée sur la représentation cachée de l'encodeur de la phrase actuelle.

$$J_t = \sum_{(S_i, S_{i+1}) \in C} -\log p(S_i | S_{i+1}) \tag{9}$$

Au moment de l'inférence, pour la génération d'un vers, chaque mot est ensuite échantillonné de manière aléatoire en fonction de la distribution de probabilité de sortie. De manière cruciale, le décodeur est entraîné à prédire les mots de la phrase suivante en sens inverse, de sorte que le dernier mot du vers soit le premier généré. Cette opération inverse est importante pour une incorporation efficace de la rime, comme cela sera expliqué dans la section suivante. Une représentation graphique de l'architecture, qui inclut les contraintes discutées ci-dessous, est donnée dans le figure 1.

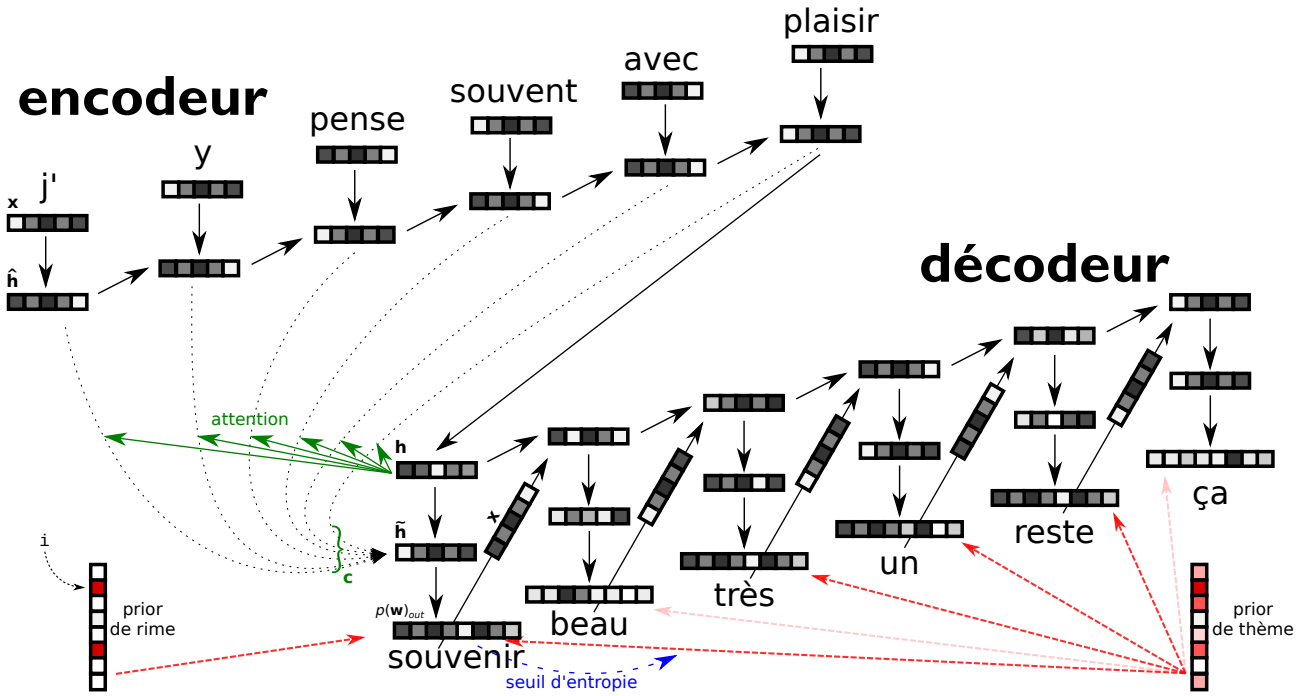

FIGURE 1: Représentation graphique du modèle de génération de poésie. L'encodeur traite le vers actuel mot par mot, et la représentation finale est donnée au décodeur, qui prédit le vers suivant mot par mot, à l'envers. Le mécanisme d'attention est représenté pour le premier pas de temps. La distribution de probabilité *a priori* pour le rime est appliquée au premier pas de temps, et celle pour le thème est facultativement appliquée à tous les pas de temps, en fonction de la valeur d'entropie de la distribution de sortie du réseau.

3.2 Contraintes poétiques comme distributions *a priori*

Étant donné que l'architecture neuronale ci-dessus est entraînée sur des textes génériques, sa sortie ne ressemblera en rien à un poème ; afin de doter la sortie générée d'un certain caractère poétique, on modifiera la distribution de probabilité de sortie du réseau de neurones en appliquant une distribution de probabilité *a priori*. On modélisera deux types de contraintes : une contrainte de rime et une contrainte thématique.

3.2.1 Contrainte de rime

Pour la modélisation de la contrainte de rime, on s'appuie sur une représentation phonétique des mots, extraite de manière automatique depuis le *Wiktionnaire* pour le français. Pour chaque mot, on détermine son rime (c'est-à-dire le groupe de voyelles final, éventuellement suivi d'un groupe de consonnes), ainsi que la groupe de consonnes précédente. Un échantillon de rimes ainsi extraites est donné dans le tableau 1.

mot	rime
reproduit	('dɥ', 'i')
thérapie	('p', 'i')
examen	('m', 'ɛ̃')
canadien	('dj', 'ɛ̃')

TABLE 1: Exemples de rimes extraits du *Wiktionnaire*

L'étape suivante consiste à créer une distribution de probabilité *a priori* pour un son de rime requis :

$$p(\mathbf{w})_{rime} = \frac{1}{Z}\mathbf{x} \text{ avec } \begin{cases} x_i = 1 & \text{if } i \in R \\ x_i = \varepsilon & \text{otherwise} \end{cases} \tag{10}$$

où R est l'ensemble des mots avec le son de rime requis, ε est une valeur très petite pour éviter les erreurs de calcul, et Z est une constante de normalisation pour assurer une distribution de probabilité. On est maintenant en mesure d'appliquer la distribution de probabilité *a priori* afin de repondérer la distribution de probabilité de sortie du réseau de neurones selon la formule 11, chaque fois que le schéma de rimes le requiert :

$$p(\mathbf{w})_{out} = \frac{1}{Z}(p(\mathbf{w}^t|w^{<t}) \odot p(\mathbf{w})_{rime} \tag{11}$$

avec $\odot$ étant la multiplication élément par élément. Rappelons que chaque vers est généré à l'envers ; la repondération par rapport à la rime est appliquée tout au début de la génération, et le mot rime est généré en premier. Cela empêche la génération d'un mot rime maladroit qui ne correspond pas au reste du vers.

3.2.2 Contrainte thématique

Pour la modélisation de la contrainte thématique, on s'appuie sur un modèle de sémantique latente sous forme d'une factorisation en matrices non négatives (NMF ; Lee & Seung, 2001). Des recherches antérieures ont montré que la méthode est capable de produire des dimensions thématiques bien

claires et interprétables (Murphy *et al.*, 2012). Comme entrée, on construit une matrice de fréquence **A**, qui capture les fréquences [1] de co-occurrence des mots du vocabulaire et leurs contextes. Cette matrice est alors factorisée en deux autres matrices non négatives, **W** et **H**.

$$\mathbf{A}_{i\times j} \approx \mathbf{W}_{i\times k}\mathbf{H}_{k\times j} \tag{12}$$

où k est beaucoup plus petit que i, j, de manière que les instances et les traits sont exprimés par un nombre limité de dimensions. De manière cruciale, la factorisation en matrices non négatives impose la contrainte que les trois matrices doivent être non négatives, c'est-à-dire tous les éléments doivent être supérieurs ou égaux à zéro. En utilisant la minimisation de la divergence de Kullback-Leibler comme fonction objective, on veut trouver les matrices **W** et **H** pour lesquelles la divergence entre **A** et **WH** (la multiplication de **W** et **H**) est la plus petite. Cette factorisation est réalisée par l'application itérative de règles de mis à jour. Quelques exemples de dimensions, extraits avec la méthode, sont représentés dans le tableau 2.

dim 1	dim 20	dim 25	dim 90
tendresse	gare	hypocrisie	désespoir
joie	bus	mensonge	terrible
bonheur	métro	accuser	colère
sourires	tram	hypocrite	angoisse
baisers	rer	tort	violente
amour	tgv	arrogance	désarroi
joies	tramway	critiquer	frustration
merveilleux	autoroute	mensonges	souffrance
nostalgie	autobus	bêtises	humiliation
douceur	boulevard	reprocher	impuissance

TABLE 2: Exemples de dimensions thématiques issues de NMF (10 mots les plus saillants)

La factorisation issue du modèle NMF peut être interprétée de manière probabiliste (Gaussier & Goutte, 2005; Ding *et al.*, 2008) : la matrice **W** peut être considérée comme $p(\mathbf{w}|k)$, c'est-à-dire la probabilité d'un certain mot w du vocabulaire, étant donnée la dimension latente k. On pourrait maintenant facilement utiliser cette distribution comme une autre distribution *a priori* thématique, appliquée à chaque sortie ; cependant, une telle modification à l'aveugle de la distribution de probabilité de sortie pour chaque mot de la séquence pose des problèmes par rapport à la structure syntaxique. Pour pallier à cela, on conditionne la modification de la distribution de sortie par le calcul d'une valeur d'entropie sur cette distribution : lorsque l'entropie de la distribution de sortie est faible, le réseau de neurones connaît la choix du mot correct afin de générer une phrase bien formée, donc on ne le changera pas. En revanche, lorsque l'entropie de la distribution de sortie est élevée, on modifie la distribution en utilisant la distribution thématique $p(\mathbf{w}|k)$ d'une dimension latente comme distribution *a priori* (analogue à la formule 11), afin d'insérer la thématique souhaitée. Le seuil d'entropie, au-dessus duquel on utilise la distribution modifiée, est défini expérimentalement.

Notez que la contrainte de rime et la contrainte thématique peuvent facilement être combinées afin de générer un mot de rime thématique, en multipliant les trois distributions concernées, puis en procédant à une normalisation.

1. Les fréquences brutes sont pondérées en utilisant l'information mutuelle spécifique (*pointwise mutual information* ; Bullinaria & Levy, 2007 ; Turney & Pantel, 2010).

3.3 Cadre d'optimisation global

La génération d'un vers est réalisé dans un cadre d'optimisation global. On intègre le modèle de génération dans un cadre d'optimisation pour deux raisons. Premièrement, la génération d'un vers est un processus d'échantillonnage, sujet au hasard. Le cadre d'optimisation nous permet de choisir le meilleur échantillon en fonction des contraintes définies ci-dessus. Deuxièmement, l'optimisation nous permet de définir quelques critères supplémentaires qui aident dans la sélection du meilleur vers. Pour chaque vers final généré, le modèle génère un nombre considérable de vers candidats ; chaque candidat est alors noté en fonction des critères suivants :

— la log-probabilité du vers généré, en fonction de l'architecture encodeur-décodeur (section 3.1) ;
— respect de la contrainte de rime (section 3.2.1) ; de plus, l'extraction du groupe de consonnes précédent (cf. tableau 1) permet de donner un score plus élevé aux mots rimes avec des groupes de consonnes précédents disparates, ce qui permet d'obtenir des rimes plus intéressantes ;
— respect de la contrainte thématique (section 3.2.2) ; le score est modélisé comme la somme des probabilités de tous les mots pour la dimension définie ;
— le nombre optimal de syllabes, modélisé comme une distribution gaussienne avec une moyenne μ et un écart-type σ ; [2]
— la log-probabilité d'un modèle de n-grammes standard.

Le score de chaque critère est normalisé à l'intervalle $[0, 1]$ à l'aide d'un normalisation *min-max*, et la moyenne harmonique [3] de tous les scores est considérée comme le score final de chaque candidat. Après la génération d'un nombre prédéfini de candidats, le candidat avec le score optimal est conservé et ajouté au poème.

4 Résultats et évaluation

4.1 Détails de mise en œuvre

L'architecture neuronale a été entraînée sur un corpus de textes web en français à caractère général, construit à base du corpus CommonCrawl [4]. Le corpus dans son intégralité contient 11 milliards de mots ; cependant, on effectue un certain nombre d'étapes de filtrage afin de ne conserver que des paires de phrases propres :

— on ne garde que des phrases de 20 mots maximum ;
— on ne garde que des phrases qui contiennent au moins un mot fonction (par exemple, les pronoms communs) d'une liste prédéfinie, l'idée étant de ne garder que des vraies phrases et de filtrer le bruit ;
— de toutes les phrases qui restent après les deux premières étapes de filtrage, on ne garde que les phrases qui apparaissent successivement dans un document.

2. On a également mené des expériences avec des contraintes basées sur le mètre et les pieds de vers, mais les premières expériences indiquaient que le système avait tendance à produire des vers très rigides. Un simple comptage des syllabes tend à donner une variation plus intéressante.

3. La moyenne harmonique est calculée par $\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$; elle est choisie pour balancer les différents scores.

4. `commoncrawl.org`

Après filtrage, la taille du corpus est réduite à 400 million de mots. On utilise un vocabulaire de 15 000 mots (sélectionnés par rapport à leur fréquence) ; au-delà, les mots sont remplacés par un token <unk> (dont la probabilité est fixée à zero pendant la phase de génération).

L'encodeur ainsi que le décodeur sont tous les deux constitués de deux couches de GRUs avec un état caché de 2048, et la taille de plongements de mots est de 512 ; les plongements d'encodeur, de décodeur, et de sortie sont partagés (Press & Wolf, 2017). On optimise les paramètres du modèle en utilisant une descente de gradient stochastique, partant d'un taux d'apprentissage de 0,2, qui est divisé par 4 lorsque la fonction de coût n'améliore plus sur un ensemble de validation. On utilise un *batch size* de 64, et on applique du *gradient clipping*. L'architecture neuronale a été implémentée en utilisant PyTorch (Paszke *et al.*, 2017), en s'appuyant considérablement sur le module OpenNMT (Klein *et al.*, 2017). Par rapport à la contrainte thématique, on utilise un seuil d'entropie de 2,70.

Le modèle n-gramme utilisé est un modèle standard d'ordre 3 lissé par Kneser-Ney, entrainé en utilisant *KenLM* (Heafield, 2011). Le modèle NMF est factorisé en 100 dimensions, la matrice de fréquences étant construite avec une fenêtre de phrases, et en utilisant la divergence de Kullback-Leibler comme objective. Le modèle n-gramme ainsi que le modèle NMF sont entrainés sur l'intégralité du corpus sans filtrage. Pour la contrainte de nombre de syllabes, on utilise $\mu = 12, \sigma = 2$.

On génère environ 2000 candidats pour chaque vers, selon un schéma de rimes fixe (ABAB CDCD). Quatre exemples représentatifs de poèmes générés par le système sont montré dans la figure 2. Notez qu'aucune sélection humaine n'a été effectuée sur les poèmes utilisés pour l'évaluation ; tous les poèmes ont été générés en une seule fois, sans *cherry picking*.

4.2 Évaluation humaine

L'évaluation quantitative de la créativité est loin d'être simple, et cela n'est pas moins vrai pour les artefacts créatifs qui sont générés de manière automatique. Des mesures d'évaluation automatique qui calculent la similarité de la sortie du système avec des textes de référence standard (telles que BLEU ou ROUGE), et qui pourraient être utilisés pour évaluer les tâches de génération standard, peuvent difficilement être qualifiées d'appropriées quand il s'agit de génération créative. C'est l'une des raisons pour lesquelles la plupart des chercheurs ont fait recours à une évaluation humaine, bien qu'il faille garder à l'esprit que l'évaluation de créativité textuelle est par nature subjective, en particulier en ce qui concerne la valeur poétique. Pour une discussion sur le sujet et un aperçu des différentes méthodes d'évaluation, voir Gonçalo Oliveira (2017).

Dans cette recherche, on adopte le cadre d'évaluation de Zhang & Lapata (2014), dans lequel il est demandé aux annotateurs d'évaluer les poèmes sur une échelle de cinq points, en fonction d'un certain nombre de caractéristiques, à savoir :

— *fluidité* : le poème est-il grammatical et syntaxiquement bien formé ?
— *cohérence* : le poème est-il structuré thématiquement ?
— *signification* : le poème transmet-il un message significatif au lecteur ? Le poème a-t-il un sens ?
— *caractère poétique* : le texte affiche-t-il les caractéristiques d'un poème ?

En plus, on demande aux annotateurs de juger si le poème est écrit par un humain ou un ordinateur.

Au total, on évalue six ensembles de poèmes différents, issus de différentes instanciations de modèles. Les différents ensembles de poèmes pris en compte lors de l'évaluation sont les suivants :

Malgré mon enthousiasme, le chagrin s'allonge
Le bonheur est toujours superbe
Toi, tu es un merveilleux songe
Je te vois rêver de bonheur dans l'herbe

Tu trouveras le bonheur de tes rêves
Je t'aime comme tout le monde
Je t'aime mon amour, je me lève
Je ressens pour toi une joie profonde

~

La route vers la ville est imprenable
Nous décidons de prendre le bus vers le tram
De plus la station de métro est très agréable
Je suis en train de rentrer dans ma rame

La gare, plusieurs personnes m'observent
Je suis allée dans la rue des portes
Je m'aperçois que le tgv, ça énerve
Si je la voie, c'est que la sncf est morte

~

Rien ne prouve qu'il s'indigne
Dans le cas contraire, ce n'est pas grave
Si la vérité est fausse, c'est très mauvais signe
Il est vrai que les gens le savent

Et cela est faux, mais qu'importe
En fait, le mensonge, c'est l'effroi
La négation de l'homme en quelque sorte
Le tort n'est pas de penser cela, il est magistrat

~

Hélas, après sa mort, ce fut elle qui cède
Ce fut un moment d'une angoisse extrême
Un sentiment d'incompréhension, mais sans remède
Une peur en colère, et parfois même

Un véritable sentiment de panique
Ce qui provoque une rage étrange
Il s'ensuit un drame tragique
On sent la tragédie qui, sans excès, s'arrange

FIGURE 2: Quatre exemples représentatifs de poèmes générés par le système ; les poèmes, de haut en bas, ont été générés respectivement en utilisant la dimension 1, 20, 25, et 90 (cf. tableau 2).

1. *random* : des poèmes générés par un modèle de référence aléatoire où, pour chaque vers, on sélectionne de manière aléatoire une phrase, entre 7 et 15 mots, dans un grand corpus ; l'idée est que les phrases sélectionnées par le modèle de référence seront assez fluides (puisqu'elles proviennent d'un corpus réel), mais sans cohérence (en raison de leur sélection aléatoire) ;
2. *rnn* : des poèmes générés par l'architecture neuronale décrit en section 3.1, sans aucune contrainte supplémentaire ;
3. *rime* : des poèmes générés par l'architecture neuronale, augmenté avec la contrainte de rime ;
4. nmf_{rand} : des poèmes générés par l'architecture neuronale, augmentée à la fois avec la contrainte de rime et la contrainte thématique, où l'une des dimensions NMF (induites de manière automatique) est sélectionnée de manière aléatoire ;
5. nmf_{spec} : des poèmes générés par l'architecture neuronale, augmentée à la fois avec la contrainte de rime et la contrainte thématique, où l'une des dimensions NMF (induites de manière automatique) est spécifiée manuellement [5] ;
6. *humain* : poèmes écrits par des humains [6].

22 annotateurs ont évalué 30 poèmes au total (5 pour chacun des six modèles évalués), de sorte que chaque poème a été évalué par au moins 4 annotateurs. Les résultats sont présentés dans le tableau 3.

modèle	fluidité	cohérence	signification	caractère poétique	écrit par humain (%)
random	2,95	1,86	1,68	2,18	0,00
rnn	3,45	2,73	2,59	2,55	0,27
rime	**3,82**	2,55	2,18	3,23	0,14
nmf_{rand}	3,64	3,32	3,09	2,86	0,27
nmf_{spec}	**3,82**	**3,82**	**3,55**	**3,95**	**0,45**
humain	4,59	4,59	4,50	4,81	0,95

TABLE 3: Résultats de l'évaluation humaine (score moyenne pour tous les annotateurs)

Tout d'abord, on remarque que tous les modèles fonctionnent mieux que le modèle de référence aléatoire, même en ce qui concerne la fluidité syntaxique ($p < 0,01$ en utilisant un test de permutation bilatéral ; notez que le modèle de référence est constituée de phrases réelles). Les bons scores obtenus pour nos modèles avec contraintes (*rime* et nmf_*) indiquent que l'application de contraintes ne nuit pas à la grammaticalité des vers. Deuxièmement, on constate que la contrainte de rime améliore le caractère poétique ($p < 0,05$ vis-à-vis *rnn*), et que la contrainte thématique améliore à la fois la cohérence ($p < 0,05$) et la signification ($p < 0,01$). On note également que le score pour caractère poétique est considérablement plus élevé ($p < 0,01$) pour nmf_{spec} (avec un thème qu'on pourrait juger poétique) que pour nmf_{rand} (avec un thème aléatoire, que l'on considérerait souvent comme plus banal). Finalement, on constate que les meilleurs scores par rapport à tous les critères sont obtenus avec le modèle nmf_{spec}, pour lesquels les poèmes sont jugés être écrits par un humain dans presque la moitié des cas.

5. Ceci peut être considéré comme définissant manuellement le thème du poème généré. La dimension spécifiée est sélectionnée pour son caractère poétique. On prend la dimension 1 du tableau 2 comme dimension spécifiée.

6. On prend des poèmes avec le même schéma de rimes que les poèmes générés, parmi les poèmes les mieux classés sur le site `short-edition.com`.

5 Conclusion

Dans cet article, on a présenté un système pour la génération de poésie en français. On utilise des réseaux de neurones en configuration encodeur-décodeur pour générer des vers candidats, en modifiant la distribution de sortie pour incorporer des contraintes littéraires et thématiques. Dans un cadre d'optimisation, on sélectionne alors parmi un nombre de candidats le meilleur vers pour inclusion dans le poème. Les résultats d'une évaluation humaine indiquent que le système est capable de générer des poèmes crédibles, avec des bons scores en termes de fluidité et de cohérence, ainsi qu'en termes de signification et de caractère poétique. Dans notre meilleure configuration, presque la moitié des poèmes générés sont jugés être écrits par un humain. La méthode présentée est générale, ce qui signifie qu'elle peut facilement être étendue à d'autres langues.

Afin de permettre l'utilisation et l'expérimentation ainsi que l'inspection du modèle, le système de génération de poésie est mis à disposition sous forme de logiciel open source. La version actuelle est téléchargéable en utilisant le lien `https://github.com/timvdc/poetry`.

On conclue avec quelques pistes pour des travaux futurs. Tout d'abord, on aimerait explorer différentes architectures de réseaux de neurones. Plus précisément, on pense que des approches hiérarchiques (Serban *et al.*, 2017) ainsi que le réseau dite *transformer* (Vaswani *et al.*, 2017) conviendraient particulièrement à la génération de la poésie. Deuxièmement, on aimerait incorporer d'autres dispositifs poétiques, notamment ceux basés sur le sens. La poésie captivante repose souvent sur l'utilisation d'un langage figuré, tel que le symbolisme et la métaphore. Une incorporation spécifique de tels dispositifs signifierait un pas important vers une génération de poésie vraiment inspirée.

Références

BULLINARIA J. A. & LEVY J. P. (2007). Extracting semantic representations from word co-occurrence statistics : A computational study. *Behavior research methods*, **39**(3), 510–526.

CHO K., VAN MERRIENBOER B., GULCEHRE C., BAHDANAU D., BOUGARES F., SCHWENK H. & BENGIO Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1724–1734 : Association for Computational Linguistics.

DING C., LI T. & PENG W. (2008). On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, **52**(8), 3913–3927.

GAUSSIER E. & GOUTTE C. (2005). Relation between plsa and nmf and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, p. 601–602 : ACM.

GERVÁS P. (2001). An expert system for the composition of formal spanish poetry. In *Applications and Innovations in Intelligent Systems VIII*, p. 19–32, London : Springer.

GHAZVININEJAD M., SHI X., CHOI Y. & KNIGHT K. (2016). Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 1183–1191, Austin, Texas : Association for Computational Linguistics.

GONÇALO OLIVEIRA H. (2012). Poetryme : a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, **1**, 21.

GONÇALO OLIVEIRA H. (2017). A survey on intelligent poetry generation : Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, p. 11–20.

GREENE E., BODRUMLU T. & KNIGHT K. (2010). Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, p. 524–533 : Association for Computational Linguistics.

HEAFIELD K. (2011). KenLM : faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, p. 187–197, Edinburgh, Scotland, United Kingdom.

HOPKINS J. & KIELA D. (2017). Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 168–178 : Association for Computational Linguistics.

KLEIN G., KIM Y., DENG Y., SENELLART J. & RUSH A. (2017). Opennmt : Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, p. 67–72 : Association for Computational Linguistics.

LAU J. H., COHN T., BALDWIN T., BROOKE J. & HAMMOND A. (2018). Deep-speare : A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1948–1958 : Association for Computational Linguistics.

LEE D. D. & SEUNG H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, p. 556–562.

LUONG T., PHAM H. & MANNING C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1412–1421 : Association for Computational Linguistics.

MANURUNG R., RITCHIE G. & THOMPSON H. (2012). Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, **24**(1), 43–64.

MURPHY B., TALUKDAR P. & MITCHELL T. (2012). Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, p. 1933–1950 : The COLING 2012 Organizing Committee.

OULIPO (1981). *Atlas de Littérature Potentielle*. Paris, France : Gallimard.

PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L. & LERER A. (2017). Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*.

PRESS O. & WOLF L. (2017). Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 2, Short Papers*, p. 157–163 : Association for Computational Linguistics.

QUENEAU R. (1961). *Cent mille milliards de poèmes*. Paris, France : Gallimard.

SERBAN I. V., SORDONI A., LOWE R., CHARLIN L., PINEAU J., COURVILLE A. & BENGIO Y. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*.

TURNEY P. D. & PANTEL P. (2010). From frequency to meaning : Vector space models of semantics. *Journal of artificial intelligence research*, **37**, 141–188.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł. & POLOSUKHIN I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, p. 5998–6008.

VEALE T. (2013). Less rhyme, more reason : Knowledge-based poetry generation with feeling, insight and wit. In *Proceedings of the international conference on computational creativity*, p. 152–159.

WANG Q., LUO T., WANG D. & XING C. (2016). Chinese song iambics generation with neural attention-based model. In *Proceedings of International Joint Conference on Artificial Intelligence*, p. 2943–2949.

YAN R. (2016). i, poet : Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of International Joint Conference on Artificial Intelligence*, p. 2238–2244.

ZHANG X. & LAPATA M. (2014). Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 670–680 : Association for Computational Linguistics.