Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles
Lille, France, 28 juin au 2 juillet 2021

Auto-encodeurs variationnels : contrecarrer le problème de *posterior collapse* grâce à la régularisation du décodeur

Alban Petit Caio Corro
Universite Paris-Saclay, CNRS, LISN, 91400, Orsay, France
{alban.petit,caio.corro}@limsi.fr

RÉSUMÉ

Les auto-encodeurs variationnels sont des modèles génératifs utiles pour apprendre des représentations latentes. En pratique, lorsqu'ils sont supervisés pour des tâches de génération de textes, ils ont tendance à ignorer les variables latentes lors du décodage. Nous proposons une nouvelle méthode de régularisation fondée sur le *dropout* « fraternel » pour encourager l'utilisation de ces variables latentes. Nous évaluons notre approche sur plusieurs jeux de données et observons des améliorations dans toutes les configurations testées.

ABSTRACT

Variational auto-encoders : prevent posterior collapse via decoder regularization

Variational autoencoders are powerful generative models useful to learn latent representations. However, when supervised for text generation tasks, they tend to ignore the latent variables. We propose a novel regularization method based on fraternal dropout to encourage the use of latent variables. We evaluate our approach and observe improvements in all the tested configurations.

MOTS-CLÉS : auto-encodeurs variationnels, régularisation, génération automatique de textes.

KEYWORDS: variational auto-encoders, regularization, automatic text generation.

1 Introduction

Les modèles génératifs reposant sur une pondération neuronale comme les auto-encodeurs variationnels (Kingma & Welling, 2014, AEV) et les réseaux antagonistes (Goodfellow *et al.*, 2014, RA), entre autres, connaissent une grande popularité dans tous les domaines de l'apprentissage automatique dont le traitement automatique des langues (TAL). Les AEV peuvent manipuler des variables observées discrètes ce qui les rend particulièrement intéressants pour la génération de textes, contrairement aux RA. Dans ces modèles, la génération d'une phrase suit le processus suivant :

$$\boldsymbol{z} \sim p(\boldsymbol{z}), \quad x_1 \sim p_\theta(x_1|\boldsymbol{z}), \quad x_2 \sim p_\theta(x_2|\boldsymbol{z}, x_1) \quad x_3 \sim p_\theta(x_3|\boldsymbol{z}, x_1, x_2), \quad \dots$$

où $\boldsymbol{z} \in \mathbb{R}^k$ est une représentation (ou plongement) de phrase latente et $x_1, x_2, \cdots \in X$ les mots composant la phrase tirés d'un vocabulaire X. La génération se termine lorsqu'un mot spécial indiquant la fin de la phrase est généré. Sans perte de généralité, nous faisons l'hypothèse que la loi *a priori* $p(\boldsymbol{z})$ est pré-définie et non apprise. L'indice θ dénote les paramètres de la distribution conditionnelle et dans notre cas, θ correspond aux paramètres d'un réseau de neurones. Il est important de noter que nous ne faisons aucune hypothèse d'indépendance dans la distribution conditionnelle $p(x_t|\boldsymbol{z}, \boldsymbol{x}_{<t})$.

Volume 1 : conférence principale, pages 2–10.

Lors de la phase d'apprentissage, l'objectif est de calculer la valeur des paramètres θ pour maximiser la vraisemblance des données :

$$\max_{\theta} \mathbb{E}_{\tilde{p}(\boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x})\right] = \max_{\theta} \mathbb{E}_{\tilde{p}(\boldsymbol{x})} \left[\int \log p_\theta(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}\right] = \max_{\theta} \mathbb{E}_{\tilde{p}(\boldsymbol{x})} \left[\mathcal{L}(\boldsymbol{x}, \theta)\right] \quad (1)$$

où $\tilde{p}$ est la distribution empirique des données d'entrainement. L'objectif est incalculable dans le cas général, sans même parler d'optimiser celui-ci, à cause de la marginalisation sur la représentation latente $\boldsymbol{z}$. Les méthodes variationnelles proposent d'introduire une loi de proposition $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ et de construire un objectif de substitution comme suit :

$$\mathcal{E}(\boldsymbol{x}, \theta, \phi) = \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]}_{\text{reconstruction}} - \underbrace{\mathrm{KL}[q_\phi(\cdot|\boldsymbol{x})|p(\cdot)]}_{\text{divergence avec l'}\textit{a priori}} \quad (2)$$

où une borne variationnelle (l'*evidence lower bound* ou ELBO), notée $\mathcal{E}$, peut être utilisée pour optimiser une borne sur l'objectif de l'équation 1 car $\forall \phi : \mathcal{E}(\boldsymbol{x}, \theta, \phi) \leq \mathcal{L}(\boldsymbol{x}, \theta)$. L'objectif de l'entraînement devient alors :

$$\max_{\theta, \phi} \mathbb{E}_{\boldsymbol{x} \sim \tilde{p}(\boldsymbol{x})} \left[\mathcal{E}(\boldsymbol{x}, \theta, \phi)\right] \quad (3)$$

où l'optimisation se fait à la fois sur les paramètres θ et sur les paramètres de la distribution de proposition ϕ. L'algorithme *Expectation-Maximization* (Dempster *et al.*, 1977, EM) résout ce problème en optimisant l'objectif par blocs, c'est à dire en optimisant successivement l'objectif en fonction de ϕ (étape E) et θ (étape M). Contrairement à EM, l'approche dite des AEV consiste à optimiser ce problème par montée de gradient stochastique jointe sur ϕ et θ. De plus, contrairement aux applications standards d'EM, ni la distribution ni la famille de la *posterior* $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ ne sont connues. Il est donc fait l'hypothèse d'indépendance sur les coordonnées de $\boldsymbol{z}$ dans la distribution q_ϕ. Enfin, la distribution q_ϕ est paramétrée par un réseau de neurones et apprise sur l'ensemble des données. Notons que lors de l'entrainement, le terme de reconstruction dans l'équation 2 est approximé via Monte-Carlo en utilisant un seul échantillon. Il est donc usuel d'appeler la distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ qui génère une représentation latente à partir d'une phrase comme l'**encodeur** et la distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ qui reconstruit la phrase original comme le **décodeur**.

Malheureusement, en pratique, les AEV pour la génération automatique de textes sont sensibles au phénomène d'effondrement de l'*a posteriori* (Bowman *et al.*, 2016, *posterior collapse*). Informellement, cela signifie que la loi de proposition n'est pas optimisée correctement et reste proche de la distribution *a priori* pour tous les points : $\forall \boldsymbol{x} : q_\phi(\boldsymbol{z}|\boldsymbol{x}) \simeq p(\boldsymbol{z})$. Cela conduit à une très mauvaise approximation de l'objectif 1 par la ELBO. *In fine*, le décodeur va ignorer cette variable latente et donc aucune représentation de phrase $\boldsymbol{z}$ n'est apprise.

De nombreux travaux se sont focalisés sur ce problème de *posterior collapse*. Nous pouvons les diviser en deux catégories. **(1)** D'une part, des modifications de la fonction objectif ont été proposées. (Bowman *et al.*, 2016) propose d'introduire une pondération du terme de divergence qui permet de tempérer son importance durant l'entrainement. (Kingma *et al.*, 2016) et (Pelsmaeker & Aziz, 2020) ont proposé d'ajouter des contraintes sur le terme de divergence qui obligent ce dernier à être supérieur à un hyperparamètre. Enfin, d'autres travaux ont proposé des fonctions de pertes alternatives pour entrainer des AEV dans le cadre de la génération de textes (Livne *et al.*, 2020; Havrylov & Titov, 2020), entre autres. **(2)** D'autre part, des travaux ont proposé de modifier l'architecture du décodeur. (Yang *et al.*, 2017) ont proposé de remplacer le réseau de neurones récurrents dans le décodeur par un réseau convolutif. (Dieng *et al.*, 2019) proposent d'utiliser des connexions *skip* entre la représentation latente et les différentes couches cachées du décodeur. Ces connexions forment des

« raccourcis » entre la variable latente et les couches de sorties afin de promouvoir cette première. L'idée est d'utiliser des architectures neuronales qui se reposent davantage sur la variable latente et donc amoindrissent indirectement l'impact de la divergence avec la distribution *a priori* sur l'objectif.

Dans ce travail, nous proposons une nouvelle approche pour contrecarrer le problème d'effondrement de l'*a posteriori*, fondée sur la régularisation des décodeurs, c'est à dire que nous ne modifions ni la fonction objectif ni la structure du décodeur. Nos contributions peuvent être résumées comme suit :

— Nous proposons d'utiliser la régularisation des paramètres pour contrecarrer l'impact de l'effondrement de l'*a posteriori*. Nous nous focalisons sur le *dropout* « fraternel » (Zolna *et al.*, 2018) pour obliger le décodeur à utiliser la représentation latente.

— Nous expérimentons notre approche dans différentes configurations en utilisant les travaux de (Li *et al.*, 2019) comme point de référence. Nous soulignons que nous ne changeons pas les hyperparamètres de leur modèle et les réutilisons tels que distribués dans le code source pour ne pas biaiser les résultats en faveur de notre approche.

Nous espérons que cet article va encourager de futurs travaux à explorer cette nouvelle piste d'amélioration pour la génération automatique de textes fondée sur les AEV.

2 Régularisation du décodeur : le *dropout* « fraternel »

Afin d'éviter le problème d'effondrement de la distribution *a posteriori*, une idée est d'augmenter artificiellement l'importance du gradient de l'encodeur provenant du décodeur. C'est cette intuition qui est utilisée dans l'architecture proposée par (Dieng *et al.*, 2019). Le LSTM (Hochreiter & Schmidhuber, 1997), un réseau neuronal récurrent, est une architecture efficace qui peut minimiser le terme de reconstruction tout en ignorant la valeur de la variable latente. C'est notamment l'objectif qui est atteint par les modèles de langues à l'état de l'art. Notre approche consiste à régulariser les paramètres du LSTM afin que les représentations cachées à chaque étape soient moins dépendantes des mots donnés en entrée. Dans ce cas, le décodeur est forcé d'utiliser la représentation cachée, donnant lieu à un gradient plus important à l'encodeur. Pour cela, nous proposons d'utiliser le *dropout* « fraternel » (Zolna *et al.*, 2018).

Le terme de reconstruction dans la fonction objectif des AEV est un terme de maximisation de la log-vraisemblance utilisant la technique habituelle de supervision forcée pour les modèles de langue : lors de l'apprentissage, ce modèle autorégressif est entrainé à prédire le mot suivant en fonction de la séquence de mots précédemment observée dans les données. Soit $\boldsymbol{x} \in X^n$ une phrase de longueur n. Chaque mot x_i est représenté par un vecteur provenant d'une table de plongements lexicaux. Nous désignons l'ensemble de ces vecteurs par une matrice $\boldsymbol{E} \in \mathbb{R}^{w \times n}$ où w est la dimension des plongements lexicaux. Une représentation contextuelle est calculée pour chaque position en utilisant le LSTM :

$$\boldsymbol{H} = \textsc{Lstm}(\boldsymbol{E}, \boldsymbol{z}; \theta)$$

où $\boldsymbol{H} \in \mathbb{R}^{d \times n}$ est une matrice contenant les états cachés de dimension d en sortie du LSTM paramétré par θ, pour chaque mot de la phrase. La variable latente $\boldsymbol{z}$ est projetée puis donnée à la fois comme initialisation de la mémoire et de l'état caché. Elle est également concaténée à chaque entrée. Nous reportons le lecteur à (Li *et al.*, 2019) pour plus de détails sur l'architecture et les différents paramètres.

Le *dropout* de plongements lexicaux (Dozat & Manning, 2017) consiste à remplacer aléatoirement certains plongements par un vecteur de 0 lors de l'entrainement pour éviter le sur-apprentissage. Le

calcul des représentations cachées devient alors :

$$\boldsymbol{d} \sim \mathcal{B}(b), \qquad \boldsymbol{E}' \in \mathbb{R}^{w \times n} \text{ t.q. } E'_{i,j} = E_{i,j} d_j, \qquad \boldsymbol{H} = \textsc{Lstm}(\boldsymbol{E}', \boldsymbol{z}; \theta)$$

où $\boldsymbol{d} \in \{0,1\}^n$ est un vecteur de booléens dont chaque élément est tiré indépendemment d'une Bernoulli de paramètre $b \in [0,1]$. La matrice $\boldsymbol{E}'$ correspond à la matrice $\boldsymbol{E}$ où les éléments de certaines colonnes ont été remplacés par des 0. Le *dropout* « fraternel » consiste quant à lui à créer deux matrices $\boldsymbol{E}'$ et $\boldsymbol{E}''$ de la façon suivante :

$$\begin{aligned}
&\boldsymbol{d} \sim \mathcal{B}(b), \\
&\boldsymbol{E}' \in \mathbb{R}^{d \times n} \text{ t.q. } E'_{i,j} = E_{i,j} d_j, && \boldsymbol{E}'' \in \mathbb{R}^{d \times n} \text{ t.q. } E''_{i,j} = E_{i,j}(1 - d_j), \\
&\boldsymbol{H}' = \textsc{Lstm}(\boldsymbol{E}', \boldsymbol{z}; \theta), && \boldsymbol{H}'' = \textsc{Lstm}(\boldsymbol{E}'', \boldsymbol{z}; \theta).
\end{aligned}$$

Les matrices $\boldsymbol{E}'$ et $\boldsymbol{E}''$ sont ensuite utilisées pour calculer deux fois la log-vraisemblance de la phrase, leur moyenne remplaçant le terme initial dans le terme de reconstruction. Un terme de régularisation est ensuite ajouté à la fonction objectif présentée dans l'équation 3 :

$$\mathcal{R}(\theta; \alpha) = -\alpha \| \textsc{Lstm}_\theta(\boldsymbol{E}') - \textsc{Lstm}_\theta(\boldsymbol{E}'') \|_2^2$$

où $\alpha > 0$ est un hyperparamètre. Pour que ce terme soit maximisé, le modèle devra utiliser l'information commune aux deux décodages, c'est à dire la représentation latente $\boldsymbol{z}$.

3 Expériences

Pour évaluer notre solution, nous avons utilisé le code source et les métriques implémentés par (Li *et al.*, 2019) auquel nous avons ajouté notre méthode de régularisation. Nous avons conservé les mêmes hyperparamètres que ceux distribués par les auteurs pour ne pas biaiser les résultats en notre faveur.

Nous nous évaluons sur deux jeux de données : Yelp (Shen *et al.*, 2017) et les données Stanford Natural Language Inference (Bowman *et al.*, 2015, SNLI). Ces deux jeux de données ont été sous-échantillonnés pour contenir 100 000 phrases d'entrainement et 10 000 pour la validation et le test chacun. SNLI et Yelp ont des vocabulaires respectivement de taille 9 990 et 8 411 et ont 10 mots par phrase en moyenne.

3.1 Métriques d'évaluation

Nous utilisons différentes métriques pour évaluer la qualité des modèles probabilistes appris.

(Log-vraisemblance et perplexité par mot) La log-vraisemblance négative (NLL) $-\mathbb{E}_{q_\phi(\boldsymbol{z}|s)} p_\theta(s|\boldsymbol{z})$ indique à quel point le modèle parvient à reconstruire l'entrée. La perplexité par mot (PPL) est la moyenne géométrique de l'inverse de la probabilité assignée au bon mot par le modèle. Puisqu'on observe l'opposé de la log-vraisemblance dans le premier cas et l'inverse des probabilités dans le second, on souhaite minimiser ces deux valeurs. Ces deux métriques sont approximés via 100 échantillons tirés de la distribution q pour chaque phrase.

(Score BLEU) Pour chaque phrase du jeu de test, une représentation latente est échantillonnée dans la distribution q puis une phrase est générée sans guider le modèle à partir de cette représentation. Le

α	NLL ↓	PPL ↓	UA ↑	IM ↑	BLEU ↑
0.01	**28.91**	**18.44**	4	6.34	5.20
0.1	29.50	19.12	5	7.03	6.40
0.5	30.69	21.53	**6**	**7.47**	**6.81**
1.0	32.30	25.28	4	6.87	6.03
2.0	33.41	28.27	4	7.29	6.09

TABLE 1 – Impact de l'hyperparamètre α du *dropout* fraternel sur les données Yelp. On veut maximiser les valeurs suivies du symbole ↑ et minimiser celles suivies par ↓.

score BLEU (Papineni *et al.*, 2002) va représenter la proportion de n-grammes (pour n allant de 1 à 4) de la phrase générée que l'on retrouve effectivement dans la phrase source. Si la phrase générée est plus courte que la phrase source, une pénalité est appliquée car il est plus simple de ne pas faire d'erreurs quand moins de mots sont générés.

(Unités actives) Le nombre d'unités actives (UA) indique le nombre de dimensions de la variable latente qui co-varient avec les observations. D'après (Burda *et al.*, 2016), un plus grand nombre d'unités actives est généralement représentatif d'une représentation latente plus riche. Comme dans leur article, une unité est considérée active si $Cov(s, \mathbb{E}_{q_\phi(\boldsymbol{z}|s)}[\boldsymbol{z}]) > 0.01$. Dans la suite de cet article, la dimension des variables latentes étant 32, nous aurons au maximum 32 unités actives.

(Information mutuelle) Nous reportons également l'information mutuelle entre la variable latente et la distribution de sortie. Une information mutuelle plus élevée indiquera que la variable latente est mieux utilisée par le modèle. Nous suivons la méthodologie de (He *et al.*, 2019).

3.2 Baseline

Nous évaluons notre approche en la comparant à plusieurs références, incluant un AEV « standard ». Toutes nos expériences utilisent une pondération du terme de divergence lors de l'optimisation du modèle. Cette technique proposée par (Bowman *et al.*, 2016) consiste à faire varier progressivement le facteur de pondération de 0 jusqu'à 1 lors des premières itérations complètes de l'entraînement. Cela impose au modèle de ne pas tenir compte de ce terme pendant les premières étapes de l'apprentissage, qui se focalisent donc sur l'apprentissage de la représentation.

Bits gratuits (Kingma *et al.*, 2016) La technique des bits gratuits consiste à ajouter une contrainte sur le terme de divergence avec l'*a priori* pour que celui-ci ne tombe pas en dessous d'une valeur pré-définie λ. Nous fixons $\lambda = 8$ dans toutes nos expériences.

Pré-entrainement (Li *et al.*, 2019) Cette solution propose d'entrainer d'abord le modèle comme un auto-encodeur standard. Ensuite, le décodeur est réinitialisé puis le modèle est entrainé comme un AEV.

3.3 Résultats et analyses

Nous reportons l'impact de l'hyperparamètre α du *dropout* « fraternel » sur les données Yelp dans la table 1. Nous observons qu'il existe un compromis entre les différentes métriques d'évaluation. Il s'agit donc ici de trouver un point d'équilibre entre la minimisation de la NLL et de la PPL et la maximisation des UA, de l'IM et du score BLEU. Dans la suite des expériences, nous avons fixé $\alpha = 0.1$ qui semble être une valeur raisonnable.

	Yelp					SNLI				
Configuration	NLL ↓	PPL ↓	UA ↑	IM ↑	BLEU ↑	NLL ↓	PPL ↓	UA ↑	IM ↑	BLEU ↑
Standard	33.40	28.25	2	1.14	1.43	32.57	20.64	**3**	0.52	2.32
+ dropout fraternel	**29.50**	**19.12**	**5**	**7.03**	**6.40**	**30.01**	**16.28**	2	**4.75**	**5.73**
Bits gratuits	29.54	19.20	32	5.69	4.02	28.88	14.66	32	4.63	4.77
+ dropout fraternel	**25.46**	**12.76**	32	**8.65**	**11.23**	**27.92**	**13.40**	32	**7.11**	**8.44**
Pré-ent.	33.74	29.21	2	0.71	0.83	31.76	19.14	3	1.14	2.76
+ dropout fraternel	**26.18**	**13.71**	**22**	**8.24**	**9.69**	**24.69**	**9.92**	**22**	**8.32**	**13.43**
Bits gratuits + Pré-ent.	25.93	13.37	32	8.14	7.54	23.33	8.75	32	8.49	13.9
+ dropout fraternel	**23.63**	**10.62**	32	**8.81**	**13.54**	**21.00**	**7.04**	32	**9.07**	**21.35**

TABLE 2 – Résultats des différentes métriques sur Yelp et SNLI pour quatres variantes de VAE avec et sans *dropout* fraternel. On veut maximiser les valeurs suivies du symbole ↑ et minimiser celles suivies par ↓.

Bits gratuits + pré-ent.
a boy is in front of a group of people.
a man in a blue shirt is standing in front of a crowd of people.
a child in blue is holding a camera.
a child in blue pants holding a camera while another man watches.
a child in blue pants holding a camera while another man in a black shirt looks on.
Bits gratuits + pré-ent. + dropout fraternel
the young boy is in a picture.
the young child is in front of a mother.
the small child is in front of a mother.
a small child in pink holds a picture of her mother.
a small child in pink sits in a picture with her mother.

TABLE 3 – Exemples d'interpolations entre deux représentations échantillonnées *a priori* pour les configurations avec un pré-entrainement et les bits gratuits sur SNLI. La deuxième configuration inclue aussi le dropout fraternel.

Nous reportons les résultats de quatre configurations de (Li *et al.*, 2019) avec et sans le *dropout* fraternel » dans le tableau 2. Notre approche apporte des améliorations pour toutes les métriques pour tous les modèles sur les deux jeux de données. Le phénomène d'effondrement de l'*a posteriori* est important sur les configurations n'utilisant ni bits gratuits ni *dropout* « fraternel », l'information mutuelle étant autour de 1 et le nombre d'unités actives étant de 2. L'ajout du *dropout* « fraternel » permet de gagner entre 4 et 7 points d'information mutuelle dans les deux configurations. Dans la configuration où le modèle est pré-entrainé, on observe que le pré-entrainement seul n'empêche pas l'effondrement de la distribution *a posteriori* puisqu'on voit 2 et 3 UA respectivement alors que le modèle conserve 22 unités actives avec le *dropout* « fraternel ». De façon intéressante, notre approche a un impact supérieur en comparaison aux bits gratuits pour toutes les métriques sauf le score UA sur Yelp ainsi que sur l'IM et le score BLEU sur SNLI : ceci peut indiquer que cette dernière approche force artificiellement les variables latentes à être décorrélées les unes des autres sans pour autant avoir l'impact escompté sur les autres métriques, car le *dropout* « fraternel » atteint les mêmes mesures avec seulement 5 et 2 unités actives respectivement sur Yelp et SNLI.

Comme expliqué précédemment, le score BLEU est calculé sur des phrases générées sans guider le modèle et permet donc aussi d'estimer la qualité des représentations latentes. Encore une fois, ce score est significativement meilleur lorsque l'on introduit le *dropout* « fraternel ».

Interpolation Nous reportons sur la Table 3 des exemples de phrases générées via interpolation entre deux échantillons de l'*a priori*. Nous observons que notre méthode semble générer des phrases cohérentes avec une évolution progressive de la longueur et du sens entre les différentes phrases.

4 Conclusion

Dans cet article, nous proposons de régulariser le décodeur pour contrecarrer le problème de l'effondrement de l'*a posteriori* lors de l'entrainement d'un AEV. Cette approche est différente de ce qui a été exploré dans la littérature. Nous observons qu'elle atteint ses deux objectifs qui sont d'améliorer la qualité de la génération de texte et surtout d'accroître l'utilisation de la variable latente. Des travaux futurs pourront s'intéresser à l'utilisation d'autres méthodes de régularisation des réseaux de neurones récurrents (Kanuparthi *et al.*, 2019; Krueger *et al.*, 2016; Gal & Ghahramani, 2016).

Remerciements

Nous remercions les 3 relecteurices anonymes pour leurs remarques et suggestions. Nous remercions François Yvon et Matthieu Labeau pour les relectures. Ces travaux ont bénéficié de calculs réalisés sur la plateforme Saclay-IA et d'un accès aux moyens de calcul de l'IDRIS au travers de l'allocation de ressources 20XX-AD011011600 attribuée par GENCI.

Références

BOWMAN S. R., ANGELI G., POTTS C. & MANNING C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* : Association for Computational Linguistics.

BOWMAN S. R., VILNIS L., VINYALS O., DAI A., JOZEFOWICZ R. & BENGIO S. (2016). Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, p. 10–21, Berlin, Germany : Association for Computational Linguistics. DOI : 10.18653/v1/K16-1002.

BURDA Y., GROSSE R. B. & SALAKHUTDINOV R. (2016). Importance weighted autoencoders. In Y. BENGIO & Y. LECUN, Éds., *Proceedings of 4th International Conference on Learning Representations*.

DEMPSTER A. P., LAIRD N. M. & RUBIN D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society : Series B (Methodological)*, **39**(1), 1–22.

DIENG A. B., KIM Y., RUSH A. M. & BLEI D. M. (2019). Avoiding latent variable collapse with generative skip models. In K. CHAUDHURI & M. SUGIYAMA, Éds., *Proceedings of Machine Learning Research*, volume 89 de *Proceedings of Machine Learning Research*, p. 2397–2405 : PMLR.

DOZAT T. & MANNING C. D. (2017). Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* : OpenReview.net.

GAL Y. & GHAHRAMANI Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In D. LEE, M. SUGIYAMA, U. LUXBURG, I. GUYON & R. GARNETT, Éds., *Advances in Neural Information Processing Systems*, volume 29, p. 1019–1027 : Curran Associates, Inc.

GOODFELLOW I. J., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A. C. & BENGIO Y. (2014). Generative adversarial nets. In Z. GHAHRAMANI, M. WELLING, C. CORTES, N. D. LAWRENCE & K. Q. WEINBERGER, Éds., *Advances in Neural Information Processing Systems 27 : Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, p. 2672–2680.

HAVRYLOV S. & TITOV I. (2020). Preventing posterior collapse with levenshtein variational autoencoder. *arXiv preprint arXiv :2004.14758*.

HE J., SPOKOYNY D., NEUBIG G. & BERG-KIRKPATRICK T. (2019). Lagging inference networks and posterior collapse in variational autoencoders. In *Proceedings of the 7th International Conference on Learning Representations* : OpenReview.net.

HOCHREITER S. & SCHMIDHUBER J. (1997). Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

KANUPARTHI B., ARPIT D., KERG G., KE N. R., MITLIAGKAS I. & BENGIO Y. (2019). h-detach : Modifying the LSTM gradient towards better optimization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* : OpenReview.net.

KINGMA D. P., SALIMANS T., JOZEFOWICZ R., CHEN X., SUTSKEVER I. & WELLING M. (2016). Improved variational inference with inverse autoregressive flow. In D. D. LEE, M. SUGIYAMA, U. V. LUXBURG, I. GUYON & R. GARNETT, Éds., *Advances in Neural Information Processing Systems 29*, p. 4743–4751. Curran Associates, Inc.

KINGMA D. P. & WELLING M. (2014). Auto-encoding variational bayes. In Y. BENGIO & Y. LECUN, Éds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

KRUEGER D., MAHARAJ T., KRAMÁR J., PEZESHKI M., BALLAS N., KE N. R., GOYAL A., BENGIO Y., LAROCHELLE H., COURVILLE A. C. & PAL C. (2016). Zoneout : Regularizing rnns by randomly preserving hidden activations. *CoRR*, **abs/1606.01305**.

LI B., HE J., NEUBIG G., BERG-KIRKPATRICK T. & YANG Y. (2019). A surprisingly effective fix for deep latent variable modeling of text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, p. 3603–3614, Hong Kong, China : Association for Computational Linguistics. DOI : 10.18653/v1/D19-1370.

LIVNE M., SWERSKY K. & FLEET D. J. (2020). Sentencemim : A latent variable language model. *arXiv preprint arXiv :2003.02645*.

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

PELSMAEKER T. & AZIZ W. (2020). Effective estimation of deep generative language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7220–7236, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.646.

SHEN T., LEI T., BARZILAY R. & JAAKKOLA T. (2017). Style transfer from non-parallel text by cross-alignment. In I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN & R. GARNETT, Éds., *Advances in Neural Information Processing Systems*, volume 30, p. 6830–6841 : Curran Associates, Inc.

YANG Z., HU Z., SALAKHUTDINOV R. & BERG-KIRKPATRICK T. (2017). Improved variational autoencoders for text modeling using dilated convolutions. In D. PRECUP & Y. W. TEH, Éds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 de *Proceedings of Machine Learning Research*, p. 3881–3890, International Convention Centre, Sydney, Australia : PMLR.

ZOLNA K., ARPIT D., SUHUBDY D. & BENGIO Y. (2018). Fraternal dropout. In *International Conference on Learning Representations*.