Algorithmes à base d'échantillonage pour l'entraînement de modèles de langue neuronaux

Matthieu Labeau Alexandre Allauzen
LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, Campus Universitaire Bât 508, F-91405 Orsay cedex
prénom.nom@limsi.fr

RÉSUMÉ

L'estimation contrastive bruitée (NCE) et l'échantillonage par importance (IS) sont des procédures d'entraînement basées sur l'échantillonage, que l'on utilise habituellement à la place de l'estimation du maximum de vraisemblance (MLE) pour éviter le calcul du softmax lorsque l'on entraîne des modèles de langue neuronaux. Dans cet article, nous cherchons à résumer le fonctionnement de ces algorithmes, et leur utilisation dans la littérature du TAL. Nous les comparons expérimentalement, et présentons des manières de faciliter l'entraînement du NCE.

ABSTRACT

Here the title in English.

Noise Contrastive Estimation (NCE) and Importance Sampling (IS) are sampling based algorithms traditionally used to avoid computing the costly output softmax when training neural language models with Maximum Likelihood Estimation (MLE). In this work, we attempt to summarize how these procedures work, and how they have been used in the computational linguistics literature. We then compare them, and experiment with tricks that ease NCE training.

MOTS-CLÉS : Modèle de langue, Estimation contrastive bruitée, Negative Sampling.

KEYWORDS: Neural Language Model, Noise Contrastive Estimation, Negative Sampling.

Introduction

Les modèles de langue statistiques jouent un rôle essentiel dans la plupart des tâches du TAL, comme la traduction automatique et la reconnaissance de la parole. Les modèles de langue neuronaux (Bengio *et al.*, 2001; Mikolov *et al.*, 2010; Chelba *et al.*, 2014; Józefowicz *et al.*, 2016) ont prouvé leur efficacité, mais il y a un problème commun à la plupart des architectures existantes : la grande taille du vocabulaire de sortie, qui implique de très longs temps de calcul, puisque l'on normalise les scores de sortie en probabilités. Il faut aussi prendre en compte qu'il est très difficile de réaliser des prédictions sur un espace aussi grand.

Une solution de plus en plus utilisée est de réduire la taille du vocabulaire en travaillant non pas avec les mots mais avec des structures contenues dans le mot, comme les morphèmes ou les caractères. Bien que cela puisse être efficace pour certaines applications, on ne fait qu'éviter le problème. D'autres solutions communes sont de réduire ou changer la structure de la couche de sortie, à l'aide de *short-lists* (Schwenk, 2007) ou d'un *softmax hiérarchique* (Morin & Bengio, 2005; Mnih & Hinton, 2009; Le *et al.*, 2011). Les techniques de *self-normalisation* (Devlin *et al.*, 2014; Andreas *et al.*, 2015;

Chen *et al.*, 2016) permettent d'éviter cette normalisation à l'inférence, ce qui l'accélère grandement. Enfin, des méthodes basées sur l'échantillonage comme l'*échantillonage par importance* (Bengio & Sénécal, 2003; Jean *et al.*, 2015) (*Importance Sampling*), et l'*estimation contrastive bruitée* (*Noise contrastive estimation*) (Mnih & Teh, 2012) permettent d'entraîner des modèles de langue neuronaux à grand vocabulaires beaucoup plus rapidement.

Nous cherchons d'abord, dans cet article, à recenser et comparer les plus importantes de ces méthodes et leurs variantes (section 1), et à vérifier pourquoi et comment elles sont utilisées dans la littérature (section 2). Ensuite, nous évaluons ces différents algorithmes, ainsi que des heuristiques de choix des hyperparamètres, sur deux corpus de référence pour l'évaluation des modèles de langue (section 3).

Maximum de vraisemblance et normalisation

L'important temps de calcul nécessaire à l'entraînement des modèles de langue neuronaux vient habituellement de la taille du vocabulaire $\mathcal{V}$. Un modèle défini par ses paramètres θ, produit une distribution P_θ^H multinomiale sur $\mathcal{V}$, conditionné par le contexte d'entrée H. Cette distribution est construite à l'aide de la fonction *softmax* :

$$P_\theta(w|H) = \frac{e^{s_\theta(w,H)}}{\sum\limits_{w' \in \mathcal{V}} e^{s_\theta(w',H)}} = \frac{e^{s_\theta(w,H)}}{Z_\theta(H)} \tag{1}$$

Ici, la fonction $s_\theta(w, H)$, qui dépend de l'architecture du réseau, produit un score pour le couple (H, w). Le dénominateur est la *fonction de partition* $Z_\theta(H)$, qui assure que la distribution de sortie soit normalisée. Ainsi, on peut écrire chaque sortie comme le produit d'une distribution non-normalisée et de l'inverse de la fonction de partition. Habituellement, le modèle est entraîné en minimisant la log-vraisemblance négative de ces distributions conditionelles pour chaque couple $(H, w) \in \mathcal{D}$ présent dans les données.

$$NLL(\theta) = -\sum_{(H,w)\in\mathcal{D}} \log P_\theta(w|H) \tag{2}$$

Pour minimiser cet objectif, on effectue des mises à jour sur les paramètres à l'aide d'une *descente de gradient stochastique* (SGD). Ce gradient est composé de deux termes :

$$\frac{\partial}{\partial\theta} \log P_\theta(w|H) = \frac{\partial}{\partial\theta} s_\theta(w, H) - \sum_{w' \in \mathcal{V}} P_\theta(w'|H) \frac{\partial}{\partial\theta} s_\theta(w', H) \tag{3}$$

Le premier permet l'accroissement de la vraisemblance du mot w, tandis que le second permet de réduire la vraisemblance de tous les autres. Le calcul de la fonction de partition $Z_\theta(H)$ est cependant nécessaire au calcul du second terme, ainsi qu'à celui de la fonction objectif.

1 Échantilloner pour accélerer l'entraînement des modèles de langue neuronaux

Il a été démontré(Gutmann & Hyvärinen, 2013) que la fonction de partition est nécessaire au calcul de la vraisemblance d'un modèle. Paramétriser cette fonction de partition séparément des paramètres θ, comme un paramètre de "mise à l'échelle" pour chaque contexte H, ne peut pas fonctionner, puisque la minimisation de la vraisemblance pourrait se faire indépendamment des données. On s'intéresse donc aux méthodes pour l'entraînement de modèles de langue non normalisés. Les plus populaires sont basées sur l'*échantillonage par importance* (IS) et l'*estimation contrastive bruitée* (NCE).

1.1 Échantillonage par importance

L'idée, explicitée dans (Bengio & Sénécal, 2003), est de ré-écrire le second terme du gradient comme une espérance, que l'on estime à l'aide d'un échantillonage par importance, à l'aide d'une distribution à partir de laquelle on peut échantillonner facilement, P_n, on obtient l'approximation suivante pour le gradient :

$$\frac{\partial}{\partial\theta}\log P_\theta(w|H) \approx \frac{\partial}{\partial\theta}s_\theta(w,H) - \frac{1}{R}\sum_{\substack{i=1\\ \hat{w}_i\sim P_n}}^{k} r_i \frac{\partial}{\partial\theta}s_\theta(\hat{w}_i,H) \text{ avec } r_i = \frac{e^{s_\theta(\hat{w}_i,H)}}{P_n(\hat{w}_i)} \text{ et } R = \frac{1}{k}\sum_{\substack{i=1\\ \hat{w}_i\sim P_n}}^{k} r_i \tag{4}$$

ce qui est un estimateur biaisé pour le gradient de notre objectif. Un problème majeur de cette approche est que les poids r_i peuvent prendre des valeurs très élevées, ce qui augmente grandement la variance de l'estimateur. On peut réduire le biais et la variance en augmentant le nombre d'échantillons k. Cependant, cela allonge le temps de calcul. Pour éviter la croissance de la variance pendant l'entraînement, (Bengio & Sénécal, 2008) cherche à adapter la distribution auxilliaire P_n au fur et à mesure de celui-ci.

Une variante récente, le *target sampling* (Jean *et al.*, 2015), appliquée à la traduction automatique, partitionne les données et définit des sous parties du vocabulaire à partir desquellse échantillonner, qui contiennent au moins les mots cibles de leur partition. L'utilisation de probabilités uniformes sur une sous partie du vocabulaire simplifie grandement l'estimateur, et puisque les mots cibles sont présents dans la distribution, l'estimateur est cohérent. Notre objectif est ainsi équivalent à un softmax que l'on applique seulement aux mots échantillonnés.

1.2 Estimation contrastive bruitée

Le NCE, initialement décrit dans (Gutmann & Hyvärinen, 2010, 2012), permet d'estimer une densité de probabilités paramétrique à partir des données, en considérant la fonction de partition comme un paramètre séparé. Il simule l'estimation par maximum de vraisemblance en apprenant à différencier les exemples provenant des données des exemples échantillonnés d'une *distribution de bruit* auxilliaire P_n. La méthode a été appliquée aux modèles de langue par (Mnih & Teh, 2012) : pour chaque couple

$(H, w) \in \mathcal{D}$, on tire k mots de P_n. Ainsi, les probabilités conditionelles d'un mot w selon H et selon sa classe (C est à 1 si le mot vient des données, et à 0 sinon) sont les suivantes :

$$P(w|C = 1, H) = P_\theta(w|H) \text{ et } P(w|C = 0, H) = P_n(w) \tag{5}$$

ce qui nous donne les probabilités postérieures d'appartenance aux classes :

$$P(C = 1|w, H) = \frac{P_\theta(w|H)}{P_\theta(w|H) + kP_n(w)} \text{ et } P(C = 0|w, H) = \frac{kP_n(w)}{P_\theta(w|H) + kP_n(w)} \tag{6}$$

Chacune des distribution $P_\theta(w|H)$ peut-être ré-écrite comme le produit d'une distribution non normalisée $p_\theta(w|H) = e^{s_\theta(w,H)}$ et d'un paramètre Z_H dépendant de H, représentant la fonction de partition. Dans (Mnih & Teh, 2012), les auteurs remarquent qu'il est difficile d'apprendre un paramètre par contexte H possible, et que si l'on se contente de fixer ces paramètres à 1, les distributions non normalisées se normalisent d'elles-même [1]. On appelle ce processus l'*auto-normalisation*. L'objectif de classification est obtenu en maximisant la log-vraisemblance de l'appartenance de l'exemple des données w à la classe $C = 1$ et des exemples $(\hat{w}_i)_{1 \leq i \leq k}$ tirés de P_n à $C = 0$. On obtient ainsi, pour un exemple :

$$J_\theta^H(w) = \log \frac{p_\theta(w|H)}{p_\theta(w|H) + kP_n(w)} + \sum_{i=1}^{k} \log \frac{kP_n(\hat{w}_i)}{p_\theta(\hat{w}_i|H) + kP_n(\hat{w}_i)} \tag{7}$$

L'un des avantages du NCE sur l'IS est que les poids utilisés sont compris entre 0 et 1. Concernant le choix de la distribution de bruit, il est démontré(Gutmann & Hyvärinen, 2010, 2012) que l'erreur d'estimation des paramètres θ est asymptotiquement indépendante de P_n lorsque le nombre d'exemples de bruit échantillonnés par exemple venant des données k est assez grand. Ainsi, le choix de P_n revient à un compromis entre une distribution plus proche de celle des données, ou une distribution plus simple à partir de laquelle on peut facilement échantillonner, et ainsi utiliser un k grand. (Mnih & Teh, 2012) compare l'utilisation des distributions uniforme et unigramme, obtenant des résultats bien plus précis avec la seconde.

Le *BlackOut* ((Ji *et al.*, 2015)) est une approche plus récente, similaire à la fois aux NCE et IS. On peut le considérer comme un NCE avec une distribution de bruit particulière, obtenue en ajustant le poids des échantillons à l'aide de ratios similaire à ceux utilisés en IS. Comme pour le NCE, on peut utiliser des distributions non normalisées. Le principal avantage du BlackOut est que les poids utilisés introduisent une dépendance au contexte H dans la probabilité de bruit, ce qui nous donne une distribution plus proche des données à un faible coût computationnel.

1.2.1 Simplification : *Negative sampling*

La méthode du *negative sampling* (NS) a été popularisée par l'algorithme du skip-gram utilisé pour l'entraînement de word embeddings(Mikolov *et al.*, 2013), et bien que très proche du NCE, elle ne permet pas d'optimiser directement la vraisemblance d'un modèle de langue(Dyer, 2014). Cependant, (Melamud *et al.*, 2016, 2017) ont montré qu'il est tout de même viable d'entraîner un modèle de

1. Ce qui s'explique par le grand nombre de paramètres libres dans les modèles neuronaux

langue avec le NS. Comme pour le NCE, on échantillonne k exemples $(\hat{w}_i)_{1\leq i\leq k}$ d'une distribution de bruit. L'objectif est simple : il maximise la vraisemblance de la régression logistique qui différencie données du bruit. Ainsi, la fonction de score de notre modèle $s_\theta(w|H)$ ne paramétrise pas la log-densité de probabilité des données mais le log-ratio de cette densité avec celle de la distribution de bruit.

$$J_\theta^H(w) = \log \sigma(s_\theta(w|H)) + \sum_{i=1}^{k} \log \sigma(-s_\theta(\hat{w}_i|H)) \tag{8}$$

Et on obtient l'estimation de la probabilité conditionelle par la simple opération :

$$\hat{P}(w|H) \propto P_n(w) \exp s_\theta(w|H) \tag{9}$$

2 Utilisations dans la littérature

Le NCE principalement été utilisé dans le contexte de la traduction automatique : (Vaswani *et al.*, 2013; Baltescu & Blunsom, 2015) utilisent une distribution de bruit unigramme, tandis que (Zoph *et al.*, 2016) utilise une distribution de bruit uniforme. Le NCE a aussi été appliqué à la reconnaissance de la parole (Chen *et al.*, 2015), avec la distribution unigramme. Dans ces travaux, les auteurs ont fixé un paramètres représentant la log-fonction de partition à 9 (au lieu de 1) pour faciliter l'entraînement.

Malgré les garanties théoriques offertes par le NCE, (Chen *et al.*, 2016) ont démontré qu'il peut être inconsistant lorsqu'il est utilisé avec de très grands vocabulaires, avec des perplexités très différentes pour des modèles ayant convergé vers des valeurs de la fonction objectif similaires. D'autres travaux (Józefowicz *et al.*, 2016) mettent en évidence que le NCE est moins efficace que l'IS [2].

3 Comparaisons et heuristiques de paramétrage

Nous proposons de comparer les méthodes présentées ici, d'abord avec un modèle que l'on supervise attentivement, sur un corpus de taille réduite, puis avec un modèle classique sur un corpus plus grand, pour lequel maximiser la vraisemblance ne serait pas une option, car trop coûteux.

Nous faisons l'hypothèse que les difficultées liées à l'utilisation du NCE proviennent de l'auto-normalisation, qui peut être vue comme une tâche supplémentaire à effectuer en parallèle par le modèle. Nous expérimentons donc avec des heuristiques qui visent à faciliter ce processus. Ainsi, on commencera par ajouter aux méthodes présentées un modèle entraîné avec le NCE, mais dont nous normalisons le score $p_\theta(w|H) = e^{s_\theta(w,H)}$ en $P_\theta(w|H)$. Cela n'a que peu d'intérêt en pratique, mais cela nous permet d'évaluer la difficulté du modèle a s'entraîner avec une distribution non normalisée. Le résultat obtenu par le NCE normalisé peut ainsi servir de référence à laquelle comparer les heuristiques proposées.

2. Notons tout de même que l'implémentation de l'IS utilisée est légèrement différente de celle décrite dans (Bengio & Sénécal, 2003) et a des points communs avec celle du target sampling.

3.1 Heuristiques proposées

La distribution de bruit la plus couramment utilisée avec les algorithmes présentés est la distribution unigramme, que nous utiliserons ici aussi. Cependant, dans certain cas (Mikolov *et al.*, 2013), cette distribution est mise à la puissance $\alpha = 0.75$, ce qui permet de la lisser. Intuitivement, cela permet d'échantilloner plus souvent les mots plus rares, et donc de faciliter leur apprentissage. Toutefois, la distribution de bruit s'éloigne de celle des données, ce qui est censé rendre l'algorithme moins efficace.

Ensuite, suivant une idée présentée dans (Chen *et al.*, 2015), nous pouvons, au lieu de fixer le paramètre censé représenter la fonction de partition Z_c à 1, utiliser une valeur fixée plus proche de sa valeur à l'initialisation (qui dépend de la taille du vocabulaire). En pratique, cela revient à entraîner le modèle avec les distributions $p_\theta(w|H) = e^{s_\theta(w,H)}/Z_c$. Nous adoptons deux approches : d'abord, nous fixons $Z_c = |\mathcal{V}|$. Ainsi, nous aidons le modèle à s'auto-normaliser, en le plaçant à l'initialisation dans un état ou il est presque normalisé. Puis, nous apprenons la valeur de Z_c comme un paramètre du modèle. Dans ce cas, le modèle apprendra en parallèle un paramètre indépendant du contexte qui l'aidera à se normaliser tout au long de l'entraînement.

3.2 Approche expérimentale

Le corpus de taille réduite sur lequel nous comparons nos modèles est le Penn Tree Bank (PTB), muni de la totalité de son vocabulaire d'entraînement, c'est à dire environ $44K$ mots. Nous entraînons ces modèles avec $k = 100$ échantillons tirés de la distribution de bruit, lorsque cela s'applique. Nos modèles de langue sont des modèles récurrents (LSTM), ici avec 2 couches cachées de dimension 300. Les fonctions d'activations sont des *Rectified Linear Units* (Relu). Nous utilisons un algorithme de SGD classique, avec un learning rate de 1.0, et nous limitons la valeur des gradients à 5.0. Nous entraînons ces modèles pendant un minimum de 30 époques et un maximum de 60, et ne sauvegardons les époques suivantes que lorsque le modèle améliore sa performance sur l'ensemble de validation. Nous arrêtons l'entraînement lorsque que la perplexité de test n'a pas été améliorée pendant 10 époques consécutives.

Le second corpus utilisé est le WMT 1 Billion Word Benchmark (Chelba *et al.*, 2014). Les modèles sont un peu plus conséquents (la dimension de la couche cachée est 512), et nous utilisons l'optimiseur Adam (Kingma & Ba, 2014). Puisque le volume de données est beaucoup plus important, l'entraînement se fait sur entre 1 et 2 époques, Nous arrêtons ici l'entraînement lorsque que la perplexité de test n'a pas été améliorée après un nombre d'exemples équivalent au tiers d'une époque.

3.3 Résultats

Les meilleures perplexités obtenues sur les données de test des deux corpus sont présentées dans la table 1. On peut constater que les résultats obtenus avec la méthode du maximum de vraisemblance (MLE) et le NCE normalisé sont presque équivalents. Cependant, les résultats du NCE non normalisé sont de loin inférieurs, ce qui montre la difficulté pour le modèle de se normaliser lui même. Le NS est plus efficace que le NCE, mais est moins précis que l'IS, qui obtient le meilleur résultat. Lisser la distribution de bruit est efficace pour le NCE et le NS, mais ne réduit que de peu l'écart avec l'IS. Enfin, fixer ou apprendre Z_c donne de bons résultats avec le NCE. Notamment, le fait d'apprendre Z_c

Méthode	Penn TreeBank (PTB)	WMT 1B-word benchmark
MLE	150.2	-
NCE Normalisé	159.3	-
NCE	306.0	X
IS	168.3	77.9
NS	228.3	102.4
NCE + $\alpha = 0.75$	277.0	X
IS + $\alpha = 0.75$	171.5	67.2
NS + $\alpha = 0.75$	195.8	83.7
NCE + $Z_c = \|\mathcal{V}\|$	178.6	72.8
NCE + $Z_c \in \theta$	172.3	70.9

TABLE 1 – Meilleures perplexités de test obtenues respectivement sur le corpus PTB, avec un vocabulaire d'entraînement complet, et sur le WMT 1 Billion Word Benchmark, avec un vocabulaire de 64K mots. 'X' indique que l'algorithme n'a pas donné une perplexité inférieure à $|\mathcal{V}|$.

permet au NCE d'approcher le résultat obtenu avec l'IS.

Ces tendances se vérifient sur le second corpus : alors que les méthodes impliquant une normalisation sont inutilisables pour un corpus de cette taille, le NCE ne converge pas sur la durée de l'entraînement choisi. Mais le NS converge, et l'IS est ici aussi bien plus précis. Lisser la distribution améliore la perplexitée obtenue par l'IS, qui donne le meilleur résultat. Enfin, apprendre automatiquement Z_c permet ici aussi au NCE d'approcher la performance de l'IS.

4 Conclusion

Étant donné la contrainte imposée par la taille du vocabulaire sur le temps d'entraînement des modèles de langue neuronaux, nous nous sommes attachés à recenser et comparer des méthodes d'entraînement basées sur l'échantillonnage. Ces méthodes évitent l'étape coûteuse de normalisation liée à l'objectif du maximum de vraisemblance. Nous avons montré que, comme décrit dans la littérature, l'*Échantillonage par importance* est plus efficace que l'*estimation contrastive bruitée*, qui est difficile à utiliser. Cependant, nous avons montré qu'avec un paramétrage judicieux de la fonction de partition, cette dernière méthode peut se révéler très efficace.

Références

ANDREAS J., RABINOVICH M., JORDAN M. I. & KLEIN D. (2015). On the accuracy of self-normalized log-linear models. In *Advances in Neural Information Processing Systems 28 : Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, p. 1783–1791.

BALTESCU P. & BLUNSOM P. (2015). Pragmatic neural language modelling in machine translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 820–829, Denver, Colorado : Association for Computational Linguistics.

BENGIO Y., DUCHARME R. & VINCENT P. (2001). A neural probabilistic language model.

BENGIO Y. & SÉNÉCAL J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.

BENGIO Y. & SÉNÉCAL J.-S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, **19**(4), 713–722.

CHELBA C., MIKOLOV T., SCHUSTER M., GE Q., BRANTS T., KOEHN P. & ROBINSON T. (2014). One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, p. 2635–2639.

CHEN W., GRANGIER D. & AULI M. (2016). Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1975–1985, Berlin, Germany : Association for Computational Linguistics.

CHEN X., LIU X., GALES M. J. F. & WOODLAND P. C. (2015). Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*, p. 5411–5415 : IEEE.

DEVLIN J., ZBIB R., HUANG Z., LAMAR T., SCHWARTZ R. & MAKHOUL J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 1370–1380, Baltimore, Maryland : Association for Computational Linguistics.

DYER C. (2014). Notes on noise contrastive estimation and negative sampling. *CoRR*, **abs/1410.8251**.

GUTMANN M. & HYVÄRINEN A. (2010). Noise-contrastive estimation : A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, p. 297–304.

GUTMANN M. & HYVÄRINEN A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, **13**, 307–361.

GUTMANN M. & HYVÄRINEN A. (2013). Estimation of unnormalized statistical models without numerical integration. In *Proceedings of the Workshop on Information Theoretic Methods in Science and Engineering*.

JEAN S., CHO K., MEMISEVIC R. & BENGIO Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, p. 1–10, Beijing, China : Association for Computational Linguistics.

JI S., VISHWANATHAN S. V. N., SATISH N., ANDERSON M. J. & DUBEY P. (2015). Blackout : Speeding up recurrent neural network language models with very large vocabularies. *CoRR*, **abs/1511.06909**.

JÓZEFOWICZ R., VINYALS O., SCHUSTER M., SHAZEER N. & WU Y. (2016). Exploring the limits of language modeling. *CoRR*, **abs/1602.02410**.

KINGMA D. P. & BA J. (2014). Adam : A method for stochastic optimization. *CoRR*, **abs/1412.6980**.

LE H.-S., OPARIN I., ALLAUZEN A., GAUVAIN J.-L. & YVON F. (2011). Structured output layer neural network language model. p. 5524–5527, Prague, Czech Republic.

MELAMUD O., DAGAN I. & GOLDBERGER J. (2016). PMI matrix approximations with applications to neural language modeling. *CoRR*, **abs/1609.01235**.

MELAMUD O., DAGAN I. & GOLDBERGER J. (2017). A simple language model based on pmi matrix approximations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 1861–1866, Copenhagen, Denmark : Association for Computational Linguistics.

MIKOLOV T., KARAFIÁT M., BURGET L., CERNOCKÝ J. & KHUDANPUR S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, p. 1045–1048.

MIKOLOV T., SUTSKEVER I., CHEN K., CORRADO G. S. & DEAN J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. BURGES, L. BOTTOU, M. WELLING, Z. GHAHRAMANI & K. Q. WEINBERGER, Eds., *Advances in Neural Information Processing Systems 26*, p. 3111–3119. Curran Associates, Inc.

MNIH A. & HINTON G. E. (2009). A scalable hierarchical distributed language model. In D. KOLLER, D. SCHUURMANS, Y. BENGIO & L. BOTTOU, Eds., *Advances in Neural Information Processing Systems 21*, p. 1081–1088. Curran Associates, Inc.

MNIH A. & TEH Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

MORIN F. & BENGIO Y. (2005). Hierarchical probabilistic neural network language model. In R. G. COWELL & Z. GHAHRAMANI, Eds., *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, p. 246–252 : Society for Artificial Intelligence and Statistics.

SCHWENK H. (2007). Continuous space language models. *Comput. Speech Lang.*, **21**(3), 492–518.

VASWANI A., ZHAO Y., FOSSUM V. & CHIANG D. (2013). Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, p. 1387–1392, Seattle, Washington, USA : Association for Computational Linguistics.

ZOPH B., VASWANI A., MAY J. & KNIGHT K. (2016). Simple, fast noise-contrastive estimation for large rnn vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 1217–1222, San Diego, California : Association for Computational Linguistics.