Rapport projet TLNL

Leader : Adrien ZABBAN
Follower : Yanis LABEYRIE

05 novembre 2023

1 Introduction

Le but de ce projet est de coder un modèle de langage basé sur un réseau de neurone multicouche. Ce modèle de langage devra permettre de prédire le mot suivant à partir d'un contexte, qui est un ensemble de mots précédent le mot à prédire.

Pour faire cela, on utilise des embeddings. Le principe est d'associer à chaque mot un vecteur de tels sorte que deux mots similaires ont des vecteurs proches (avec une distance euclidienne), et deux mots totalement différents sont très loin. Cela permet de projeter les mots dans un espace latent pour pouvoir travailler plus efficacement sur ces mots. On note e la dimension de l'espace latent.

Dans un premier temps, on utilisera les embeddings des mots appris par l'algorithme Word2Vec [1]. Par la suite, nous tenterons d'améliorer le modèle en lui permettant d'apprendre ses propres embeddings à partir d'embeddings aléatoire, ou directement les embeddings appris de Word2Vec.

2 Modèle Initial

Nous avons implémenté le perceptron multicouche comme modèle de langage de base. Celui-ci est composé d'une couche d'entrée qui prenant un vecteur de dimension $k \times e$ représentant les embeddings de k mots concaténés. Une couche de neurone cachée dont nous avons choisi de faire varier le nombre de neurone noté h, suivit d'une fonction ReLU [2]. Puis une deuxième couche de neurones suivit d'un softmax retournant un vecteur de taille V, le nombre de mots appris. La Figure 1 représente ce modèle.

Formalisme mathématiques En notant, $W \in \mathcal{M}_{k\times e,h_1}(\mathbb{R})$, et $U \in \mathcal{M}_{h,V}(\mathbb{R})$ les matrices de poids, $(b_1, b_2) \in \mathbb{R}^h \times \mathbb{R}^V$ les biais, $X \in \mathbb{R}^{k\times e}$ le vecteur d'entrée, l'équation (1) donne la fonction de sortie $F(X) \in \mathbb{R}^V$ du modèle.

Après plusieurs essais, nous avons choisie de prendre : $k = 3$, $e = 100$, $h_1 = 256$. Et dans nos données d'entraînement, on avait un vocabulaire contenant : $V = 2908$ mots distincts. Avec ces hyperparamètres, nous avons dans ce modèle 824412 paramètres apprenables.

$$F(x) = softmax(U(ReLU(WX + b_1)) + b_2) \tag{1}$$

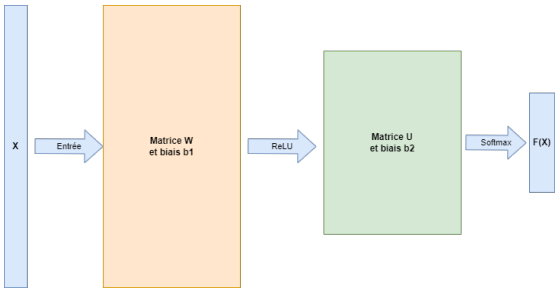


FIGURE 1 – Modèle initiale

Pour entraîner le modèle, nous avons comparer notre sortie au mot à prédire dans le format One-hot encoding [1], et nous appliquons le calcule de la fonction de coût : cross-entropy. On avons utilisé Adam [3] pour optimiser les paramètres avec un learning rate de 0.01 pour les 5 premières époques et 0.001 pour les suivantes.

Métriques Nous avons par ailleurs décidé d'évaluer ce réseau à l'aide de plusieurs métriques comme l'accuracy, la perplexité, le f_1-score et la métrique top k [2] (avec $k = 5$), voir [4].

Résultats Sur la Figure 2, on voit les courbes d'apprentissage. Les valeurs des métriques sur les données d'entraînement sont en bleue, sur les données de validations sont en orange.

2.1 Piste à creuser 1 : Apprentissage automatique des embeddings par le modèle de langage.

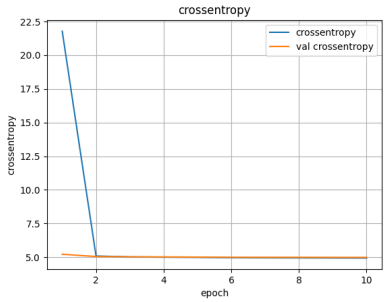
2.1.1 Description

Cette première piste que nous avons exploré consiste à se dire que notre modèle d'embeddings pré-entraîné n'est pas forcément pertinent pour la tâche de prédiction du mot suivant d'une phrase. L'idée est donc de considérer que la matrice d'embeddings peut-être apprise par le modèle de langage et que la matrice apprise sera plus pertinente qu'une matrice apprise séparément pour résoudre la tâche. On va donc considérer que les paramètres de la matrice sont des paramètres apprenables du modèle de langage et donc étendre jusqu'à eux l'algorithme de rétro-propagation du gradient.

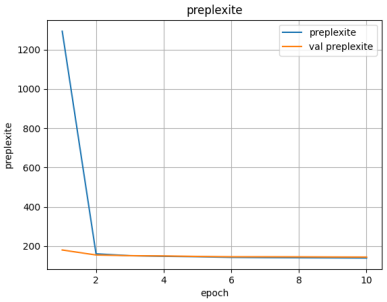
2.1.2 Mise en œuvre

Il a fallut adapter un peu notre code car avant pour avoir l'embedding d'un mot i, on utilisé $E[i]$, où $E \in \mathcal{M}_{V,e}(\mathbb{R})$ est la matrice d'embedding. Cependant,

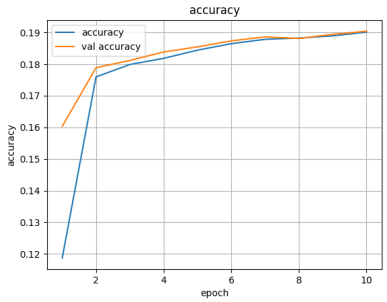
1. Le format One-hot encoding est un encodage des indices en un vecteur d'une taille du nombre d'indice possible tel que ce vecteur possède des 0 partout sauf à l'indice en question qui possède un 1.
2. La métrique "top-k" évalue un modèle en comptant combien de prédictions correctes il fait parmi les k premières prédictions les plus probables. Attention ici k n'est pas le nombre de mots dans le contexte !



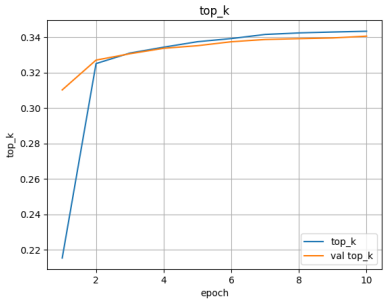
(a) Cross Entropy Loss (Lower is Better)



(b) perplexité (Lower is Better)



(c) Accuracy (Higher is Better)



(d) top k, avec k=5 (Higher is Better)

FIGURE 2 – Entraînement du modèle initiale

cette opération n'est pas dérivable, donc il a fallut transformer le mot i en un vecteur $x \in \mathbb{R}^V$ one-hot, et faire $x \times E$ pour obtenir l'embedding du mots i, qui cette fois ci, est une opération dérivable. Une fois cette adaptation faite, on a rajouté cette opération dans notre modèle et nous obtenons donc un nouveau modèle illustré par la Figure 3, et donné par l'équation (2).

$$F(x) = softmax(U(ReLU(W\tilde{X} + b_1)) + b_2) \quad (2)$$
$$\text{où } \tilde{X} = flatten(XE) \quad \text{et} \quad X \in \mathcal{M}_{k,V}(\mathbb{R})$$

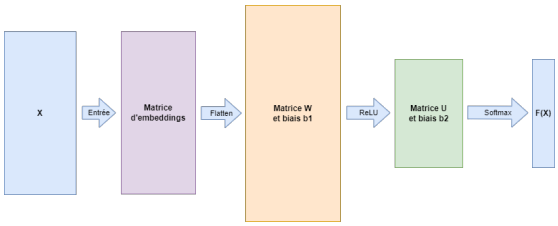


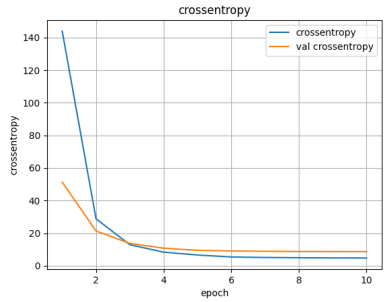
FIGURE 3 – Nouveau modèle avec l'apprentissage des embeddings

2.1.3 Résultats

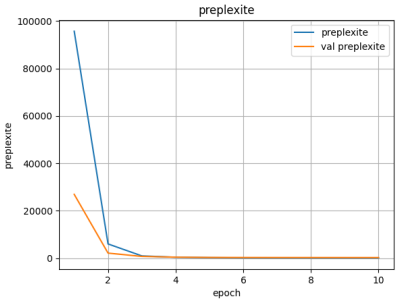
Décrire les résultats que vous avez obtenu. Dites si vos hypothèses ont été vérifiées. cf subfig :4

Si c'est le cas, donnez des exemples qui le montre.

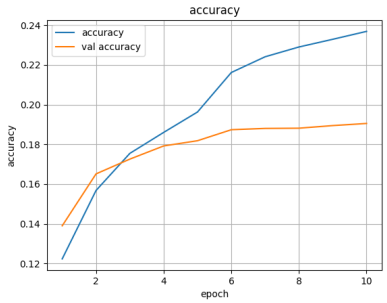
Si ce n'est pas le cas, faire une analyse des erreurs, proposez des explications et si possible mettez les en œuvre dans le cas d'une nouvelle expérience. Souvent, les choses ne marchent pas du premier coup.



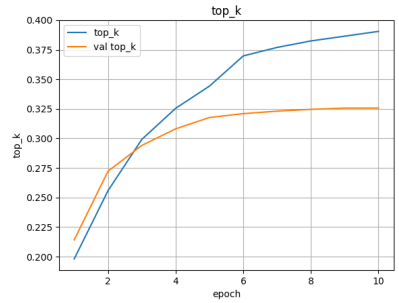
(a) Cross Entropy Loss (Lower is Better)



(b) perplexité (Lower is Better)



(c) Accuracy (Higher is Better)



(d) top k, avec k=5 (Higher is Better)

FIGURE 4 – Entraînement du modèle avec l'apprentissage des embeddings

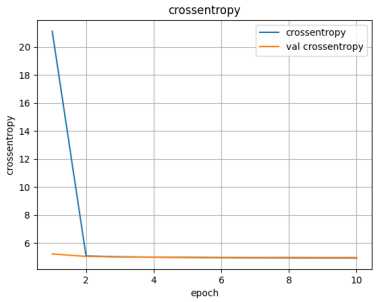
2.2 Piste à creuser 2 : titre

même structure

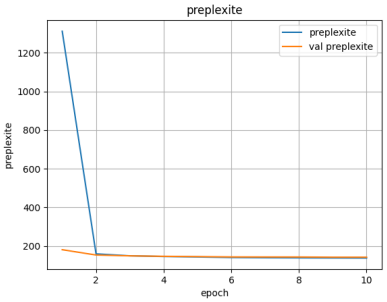
3 Conclusions et perspectives

Reprendre le problème sur lequel vous vous êtes intéressé. Décrire les hypothèses sur lesquelles vous avez travaillé. Revenez sur les résultats que vous avez obtenu et mettant en avant les points qui vous semblent intéressants.

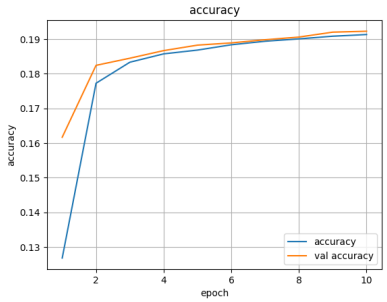
Dites à l'issue de ce travail quelle pistes pourraient être explorées pour aller plus loin.



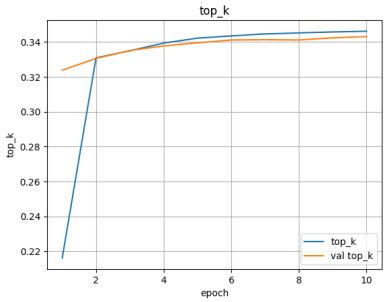
(a) Cross Entropy Loss (Lower is Better)



(b) perplexité (Lower is Better)



(c) Accuracy (Higher is Better)



(d) top k, avec k=5 (Higher is Better)

FIGURE 5 – Entraînement du modèle avec l'apprentissage des embeddings à partir des embeddings de Word2Vect

métriques	cross entropie	accuracy	top k	f_1 score	perplexité
modèle initial	4.89	0.20	0.35	$9.42e-4$	131
modèle 2	8.29	0.20	0.33	$1.10e-3$	270
modèle 3	4.89	0.20	0.35	$8.86e-4$	130

TABLE 1 – Résultat des différents modèles sur la base de données de teste

Références

[1] T. MIKOLOV, K. CHEN, G. CORRADO et J. DEAN, « Efficient estimation of word representations in vector space, » *arXiv preprint arXiv :1301.3781*, 2013.

[2] A. F. AGARAP, « Deep Learning using Rectified Linear Units (ReLU), » *CoRR*, t. abs/1803.08375, 2018. arXiv : 1803.08375. adresse : http://arxiv.org/abs/1803.08375.

[3] D. P. KINGMA et J. BA, « Adam : A method for stochastic optimization, » *arXiv preprint arXiv :1412.6980*, 2014.

[4] L. LIU, T. G. DIETTERICH, N. LI et Z. ZHOU, « Transductive Optimization of Top k Precision, » *CoRR*, t. abs/1510.05976, 2015. arXiv : 1510.05976. adresse : http://arxiv.org/abs/1510.05976.

modèles	1	2	3
temps (en s)	360	1001	460

TABLE 2 – Comparaison du temps d'entraînement des différents modèles sur 10 époques.