Les Word Embeddings à partir d'une matrice de co-occurence transformée

Approche de Factorisation matricielle

Sarah Yakoubene et Yanis Ait Hammou

- Dirigée par Lazhar Labiod

Devant un jury composé de

Lazhar Labiod, Maître de Conférences, Université de Paris, Encadrant
Severine Affeld, Maître de Conférences, Université de Paris, Encadrante

Les Word Embeddings à partir d'une matrice de co-occurence transformée

Approche de Factorisation matricielle

Sarah Yakoubene et Yanis Ait Hammou

Résumé La plupart des méthodes de word embedding existantes peuvent être classées en deux catégories : méthodes de word embedding neuronales (word2vec, gloVe et FastText), et d'autres basées sur la factorisation matricielles, généralement résolues à l'aide de la décomposition en valeurs singulières (SVD), Les tendances récentes suggèrent que les modèles de word embedding neuronales captent de nombreuses similitudes relationnelles, qui peuvent être récupérées au moyen de l'arithmétique vectorielle dans l'espace embarqué. Cependant, des études ont montré qu'une grande partie de gains de performances de word embedding sont dus à certains choix de conception et à des optimisations d'hyperparamètres, plutôt qu'aux algorithmes de word embedding eux-mêmes, il a été démontré que ces modifications peuvent être transférées vers des modèles basés sur la factorisation matricielle, et ces derniers produisant des gains similaires. D'autres études ont proposé une méthode qui surpasse le problème de perte d'information dans SVD. Nous allons réaliser une étude comparative entres ses méthodes en appliquant des transformations sur la matrice de co-occurrence, afin de mettre en lumières les différents défis posés par ce type de données aux méthodes de WE basées sur la factorisation matricielle. Pour l'évaluation des word embeddings, on se focalisera sur les deux tâches utilisées dans la littérature, qui sont la tâche de similarité et la tâche d'analogie.

Mots-clés word embedding, factorisation matricielle, SVD, PSDVec, svd2vec, apprentissage incrémental.

Abstract Most of the existing word embedding methods can be classified into two categories : neural word embedding methods (word2vec, gloVe and FastText), and others based on matrix factorization, usually solved using singular value decomposition. (SVD), Recent trends suggest that neural word embedding models capture many relational similarities, which can be retrieved using vector arithmetic in embedded space. However, studies have shown that a large part of word embedding performance gains are due to certain design choices and hyperparameter optimizations, rather than the word embedding algorithms themselves, it has been shown that these modifications can be transferred to models based on matrix factorization, and the latter producing similar gains. Other studies have proposed a method that overcomes the problem of information loss in SVD. We will carry out a comparative study between its methods by applying transformations on the co-occurrence matrix, in order to highlight the different challenges posed by this type of data to WE methods based on matrix factorization. For the evaluation of word embeddings, we will focus on the two tasks used in the literature, which are the similarity task and the analogy task.

Keywords word embedding, matrix factorization, SVD, PSDVec, svd2vec, incremental learning.

Université de Paris

Table des matières

1 Introduction **4**

2 État de l'art **6**

3 Méthodologies **8**

4 Expérimentation et Résultats **8**

4.1 Tâche de similarité 8

4.2 Tâche d'analogie 9

5 Conclusion **11**

Université de Paris

1 Introduction

Comprendre la signification d'un mot est au cœur du traitement du langage naturel (NLP). Bien qu'une compréhension profonde et humaine reste insaisissable, de nombreuses méthodes ont réussi à récupérer certains aspects de similitude entre les mots. Traditionnellement, les mots étaient remplacés par des identifiants uniques pour effectuer des calculs. Cette approche présente l'inconvénient de devoir créer une énorme liste de mots et de donner à chaque élément un identifiant unique. De plus, cette représentation perd le sens profond du mot dans une phrase. Ainsi, il perd le contexte de la phrase. Au lieu d'utiliser des nombres uniques pour nos calculs, nous pouvons également utiliser des vecteurs pour représenter leur signification, ce que l'on appelle word embedding. Le word embedding prend en compte le contexte et donne à un mot avec une signification ou une influence similaire dans une phrase une valeur similaire pour une caractéristique spécifique. Il peut contenir le contexte de la phrase à partir des mots et, après entraînement sur un grand ensemble de données, il peut même reconnaître des mots qui ne sont pas disponibles dans la représentation vectorielle à partir de phrases.

La tâche de word embedding consiste à modéliser la distribution d'un mot et de ses mots de contexte en utilisant leurs vecteurs correspondants dans l'espace euclidien. Ensuite, en effectuant une régression sur les statistiques pertinentes à partir d'un corpus, nous récupérons un ensemble de vecteurs qui correspondent le mieux à ces statistiques. Ces vecteurs, communément appelés embeddings, capturent les régularités sémantiques/syntaxiques entre les mots.

Le cœur d'une méthode de word embedding est la fonction de liaison qui relie l'entrée - les embeddings, à la sortie - certaines statistiques de corpus. Sur la base de la fonction de lien, la fonction objectif est développée. Le caractère raisonnable de la fonction de lien a un impact sur la qualité des plongements obtenus, et différentes fonctions de lien se prêtent à différents algorithmes d'optimisation, avec une évolutivité différente. Sur la base des formes de la fonction de lien et des techniques d'optimisation, la plupart des méthodes peuvent être divisées en deux classes : les modèles de word embedding neuronals et les modeles basé sur la factorisation matricielles.

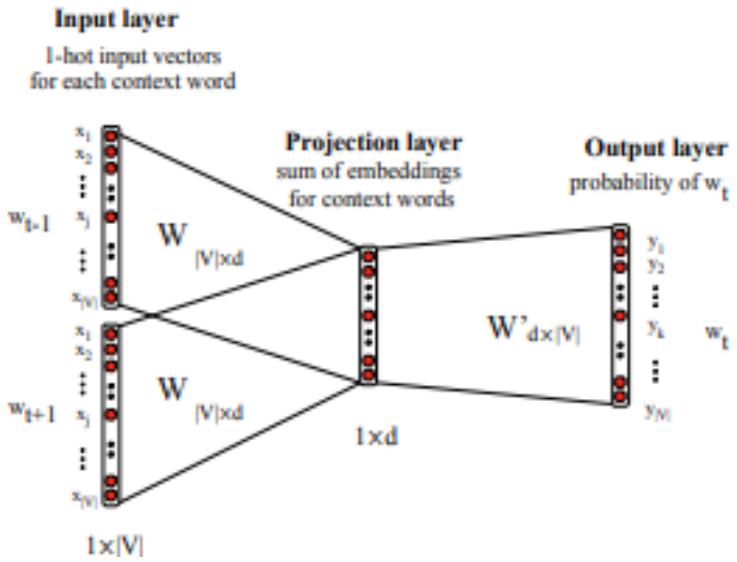

FIGURE 1 – représentation graphique de l'exemple d'architecture de modèles word embedding basés sur des réseaux de neurones.

Les modèles de word embeddings utilisent la fonction de lien softmax pour modéliser la distribution conditionnelle d'un mot en fonction de son contexte (ou vice versa) en fonction des embeddings. Le normalisateur de la fonction softmax apporte une complexité à l'optimisation, qui est généralement abordée par des méthodes basées sur le gradient. Le travail de pionnier était (Bengio et al., 2003). Plus tard, Mnih et Hinton (2007) proposent trois fonctions de liaison différentes. Cependant, il existe des matrices d'interaction entre les plongements dans tous ces modèles, ce

qui complique et ralentit l'apprentissage, les empêchant d'être entraînés sur d'énormes corpus. Mikolov et coll. (2013a) et Mikolov et al. (2013b) simplifient considérablement la distribution conditionnelle, où les deux plongements interagissent directement. Ils ont mis en œuvre le fameux «word2vec», qui peut être formé efficacement sur d'énormes corpus. Les plongements obtenus montrent d'excellents performance sur diverses tâches.

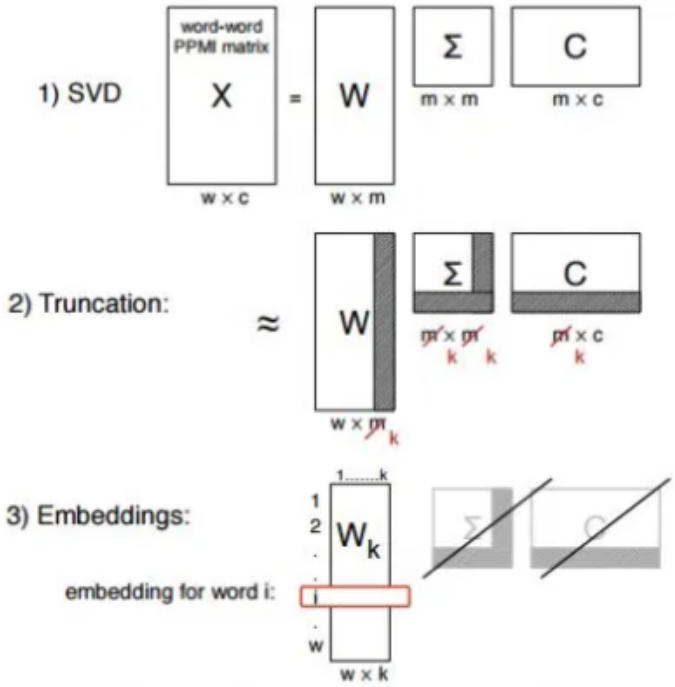

FIGURE 2 – représentation graphique de la méthode de factorisation matricielle (SVD).

Les méthodes de factorisation matricielle (MF en abrégé) comprennent diverses fonctions de liaison et méthodes d'optimisation. Les fonctions de liaison ne sont généralement pas des fonctions softmax. Les méthodes MF visent à reconstruire certaines matrices de statistiques de corpus par le produit de deux matrices de facteurs. L'objectif est généralement de minimiser l'erreur de reconstruction, éventuellement avec d'autres contraintes. Dans cette ligne de recherche, Levy et Goldberg (2014b) constatent que «word2vec» fait essentiellement du stochastique factorisation pondérée de la matrice d'informations mutuelles ponctuelles (PMI) de contexte mot. Ils factorisent ensuite cette matrice directement comme une nouvelle méthode. Pennington et coll. (2014) proposent une fonction de régression bilinéaire de la distribution conditionnelle, à partir de laquelle est formulé un problème MF pondéré sur la matrice logfréquence bigramme. Descente de Gradient est utilisé pour trouver les plongements. Récemment, sur la base de l'intuition que les mots peuvent être organisés en hiérarchies sémantiques, Yogatama et al. (2015) ajoutent des régulariseurs clairsemés hiérarchiques à l'erreur de reconstruction matricielle. Avec des techniques similaires, Faruqui et al. (2015) reconstruisent un ensemble de plongements pré-entraînés en utilisant des vecteurs clairsemés de plus grande dimensionnalité.

Notre objectif est la construction d'un modèle de word embeddings se basant sur une matrice de co-occurrences de mots, le mot est représenté par un vecteur sparse de très haute dimension, dans lequel chaque entrée est une mesure d'association entre le mot et un contexte particulier. La mesure la plus populaire de cette association est appelée : information mutuelle PMI (Pointwise Mutual Information. Ensuite, la dimension de ce vecteur est réduite en utilisant une méthode de décomposition matricielle telles que la décomposition en valeurs singulières SVD (Singular Value Decomposition).

Nous souhaitons donc aborder le problème de dérivation des Word embeddings à partir d'une matrice de co-occurrencse transformée, afin de mettre en lumières les différents challenges posés par ce type de données aux méthodes de WE basées sur la factorisation matricielle.

Egalement, il s'agira d'évaluer les word embeddings, on se focalisera sur les deux tâches large-

ment utilisées dans la littérature, qui sont la tâche de similarité et la tâche d'analogie.

2 État de l'art

On appelle la technique de représentation d'un mot, ou un ensemble de mots en vecteurs de dimension inférieure, " WE : Word Embeddings", soit littéralement "plongement de mot" . Le Word Embeddings a gagné en popularité en tant que technique importante de traitement du langage naturel (NLP pour Natural Language processing) non supervisé ces dernières années. La tâche de WE est de dériver un ensemble de vecteurs dans un espace euclidien correspondant aux mots qui correspondent le mieux à certaines statistiques dérivées d'un corpus. Ces vecteurs, Word Embeddings, capturent les régularités sémantiques et/ou syntaxiques entre les mots. Le Word Embeddings peut remplacer le codage traditionnel des mots en tant qu'entrée d'un système d'apprentissage NLP, et peut souvent améliorer considérablement les performances du système.

Il existe différentes méthodes word embeddings ; des méthodes neuronales telles que word2vec, gloVe et FastText. D'autres méthodes sont basées sur l'utilisation d'une simple décomposition matricielle telle que la SVD sur une matrice PMI (pointwise mutual information) qui donneraient des performances largement suffisante pour la plupart des applications industrielles.
L'idée fondamentale reste la même : compresser de manière non supervisée la représentation d'un mot à partir d'un gros corpus de texte représentatif, afin d'obtenir un vecteur dense qui permet de visualiser et de fournir aux algorithmes d'apprentissage des features plus intéressantes.

Embedding models
Différents modèles de word embedding donnent différentes représentations vectorielles. Il y a quelques propriétés que toutes les bonnes représentations devraient viser.

Non-confusion[1]
Différents contextes locaux autour d'un mot devraient donner lieu à des propriétés spécifiques du mot, par ex. la forme plurielle ou singulière, les temps, etc. Les modèles d'intégration devraient être capables de discerner les différences dans les contextes et de coder ces détails en une représentation significative dans le sous-espace de mots

Robustesse contre l'ambiguïté lexicale[1]
Tous les sens (ou significations) d'un mot doivent être représentés. Les modèles doivent être capables de discerner le sens d'un mot à partir de son contexte et de trouver l'incrustation appropriée. Ceci est nécessaire pour éviter les représentations dénuées de sens issues de propriétés conflictuelles qui peuvent résulter de la polysémie des mots. Par exemple, les modèles de mots devraient pouvoir représenter la différence entre les éléments suivants : «la proue d'un navire» et «l'arc et les flèches». Qui et coll[2] tentent d'améliorer le modèle d'incorporation de mots dans cette perspective.

Démonstration de multiples facettes [1]
La facette, les propriétés phonétiques, morphologiques, syntaxiques et autres d'un mot devraient contribuer à sa représentation finale. Ceci est important car les modèles de mots doivent produire des représentations de mots significatives et peut-être trouver des relations entre différents mots. Par exemple, la représentation d'un mot doit changer lorsque le temps est changé ou qu'un préfixe est ajouté.

Fiabilité[3]
Les résultats d'un modèle d'incorporation de mots doivent être fiables. Ceci est important car les vecteurs de mots sont initialisés au hasard lors de l'apprentissage. Même si un modèle crée des représentations différentes à partir du même jeu de données en raison d'une initialisation aléatoire, les performances de diverses représentations doivent obtenir un score cohérent.

Bonne géométrie[4]
La géométrie d'un espace d'encastrement doit avoir une bonne répartition. De manière générale, un plus petit ensemble de mots plus fréquents et non liés doit être réparti uniformément

dans l'espace tandis qu'un plus grand ensemble de mots rares doit se regrouper autour de mots fréquents. Les modèles de mots devraient surmonter la difficulté découlant de la fréquence incohérente de l'utilisation des mots et tirer une certaine signification de la fréquence des mots. De ce point de vue, Mu et al. et Wang et al.[5,6] des méthodes proposées pour améliorer la qualité de l'incorporation de mots en distribuant les mots de manière plus uniforme dans l'espace de grande dimension.

FastText
FastText[7]utilise les informations de sous-mot de manière explicite afin que l'incorporation de mots rares puisse toujours être bien représentée. Il est toujours basé sur le modèle skip-gram, où chaque mot est représenté comme un sac de caractères n-grammes ou d'unités de sous-mots. Une représentation vectorielle est associée à chacun des caractères n-grammes, et la moyenne de ces vecteurs donne la représentation finale du mot. Ce modèle améliore les performances sur les tâches syntaxiques de manière significative mais pas beaucoup dans les questions sémantiques.

Glove
Glove est un modèle de régression log bilinéaire globale [8] qui combine les avantages des deux grandes familles de modèles de la littérature : la factorisation matricielle globale et les méthodes de fenêtre contextuelle locale. ce modèle exploite efficacement les informations statistiques en s'entraînant uniquement sur les éléments différents de zéro dans une matrice de cooccurrence mot-mot, plutôt que sur l'ensemble de la matrice clairsemée ou sur des fenêtres de contexte individuelles dans un grand corpus. Le modèle produit un espace vectoriel avec une sous-structure significative.

Décomposition en valeurs singulières (SVD)
Bien que les représentations vectorielles clairsemées fonctionnent bien, il existe également des avantages à travailler avec des vecteurs denses de faible dimension, tels qu'une efficacité de calcul améliorée et, sans doute, une meilleure généralisation. De tels vecteurs peuvent être obtenus en effectuant une réduction de dimensionnalité sur la matrice à haute dimension clairsemée.

Une méthode courante pour ce faire est la décomposition en valeurs singulières tronquée (SVD), qui trouve la factorisation optimale de rang d par rapport à la perte L2 (Eckart et Young, 1936). Il a été popularisé en NLP via l'analyse sémantique latente (LSA) (Deerwester et al., 1990). SVD factorise M en le produit de trois matrices $U \cdot \Sigma \cdot V^T$ où U et V sont orthonormés et Σ est une matrice diagonale de valeurs propres par ordre décroissant. En ne conservant que les d éléments supérieurs de Σ, on obtient $M_d = U_d \cdot \Sigma_d \cdot V_d^T$. Les produits scalaires entre les lignes de $W = U_d \cdot \Sigma_d$ sont égaux aux produits scalaires entre les lignes de M_d.

Dans le cadre des matrices de contexte de mots, les lignes denses et d-dimensionnelles de W peuvent remplacer les lignes de très haute dimension de M. En effet, une approche courante dans la littérature NLP consiste à factoriser la matrice PPMI avec SVD, puis en prenant les rangées de :

$$W^{SVD} = U_d \cdot \Sigma_d \qquad\qquad C^{SVD} = V_d$$

sous forme de représentations de mots et de contextes, respectivement.

PMI
De nombreuses mesures de similitude sémantique de mots reposent sur des statistiques de cooccurrence de mots calculées à partir d'un grand corpus de texte, l'hypothèse est que les mots qui apparaissent fréquemment ensemble dans le texte sont liés conceptuellement. Parmi les mesures basées sur la cooccurrence de mots les plus populaires, on trouve les informations mutuelles ponctuelles (PMI). Pour deux mots (ou termes) a et b, PMI est défini comme :

$$PMI = log \frac{p(a,b)}{p(a)p(b)}$$

$p(a)$ (resp. $p(b)$) est la probabilité que le mot a (resp. b) apparaisse dans une fenêtre de texte d'une taille donnée tandis que $p(a,b)$ désigne la probabilité que a et b co-apparaissent dans la même

fenêtre. PMI est donc le log du rapport entre la fréquence de cooccurrence observée et la fréquence attendue en indépendance. Il calcule la mesure dans laquelle les mots apparaissent plus que par hasard ou sont indépendants.

3 Méthodologies

Bien que largement utilisé, PMI a deux limites principales : premièrement, il peut prendre des valeurs positives ou négatives et manque de limites fixes, ce qui complique l'interprétation. Deuxièmement, il a une tendance bien connue à donner des scores plus élevés aux événements de basse fréquence.

Afin de s'affranchir de ces limitations, plusieurs variantes de PMI ont donc été proposées au fil des années. Une étude comparative systématique et formelle se concentrant spécifiquement sur ces variantes est disponible dans [9].

Dans cette étude nous avons réalisé une comparaison entre ses méthodes : PMI, PMI^k, PMI normalisé, $PPMI$, et la méthode de décomposition factorielle (SVD) avec une variation du nombre de dimension.

PMI^k consiste à signaler un ou plusieurs facteurs de $p(a, b)$ dans le logarithme pour corriger empiriquement le biais du PMI vers les événements de basse fréquence. Les mesures PMI^2 et PMI^3 utilisées sont définies comme suit :

$$PMI^k(a, b) = PMI(a, b) - (-(k - 1)\ log\ p(a, b))$$

$NPMI$ (PMI normalisé) est proposé dans (Gerlof, 2006). La motivation principale de cette variante est de donner à PMI une borne supérieure fixe de 1 en cas de dépendance parfaite, c'est-à-dire lorsque deux mots n'apparaissent qu'ensemble.Une option pour normaliser PMI est alors de le diviser par $logp(a, b)$, ce qui donne la définition suivante :

$$NMPI(a, b) = \frac{PMI(a, b)}{-log\ p(a, b)}$$

$PPMI$ est une approche plus cohérente consiste à utiliser un PMI positif, dans lequel toutes les valeurs négatives sont remplacées par 0 :

$$PPMI(a, b) = max(PMI(a, b), 0)$$

Les autres méthodes correspondent à une transformation de PPMI via le SVD, le nombre de dimensions utilisées est de 50, 100, 300, 500 et 700.

4 Expérimentation et Résultats

Nous avons évalué chaque représentation de mot sur dix ensembles de données couvrant les tâches de similarité et d'analogie.

4.1 Tâche de similarité

Cette tâche d'évaluation est largement utilisée dans la littérature [3]. Supposons que nous disposons d'un jeu de données composé de paires de mots et d'un score de similarité attribué par des humains (score de référence) pour chaque paire. La tâche consiste à comparer les similarités entre les paires de mots à leur score de référence. D'abord, nous calculons la similarité cosinus entre les word embeddings de chaque paire de mots. Puis, nous ordonnons ces paires par leur degré de similarité. Enfin, nous calculons le coefficient de corrélation Spearman's rank entre les classements produits par les word embeddings et ceux produits par les scores de référence.

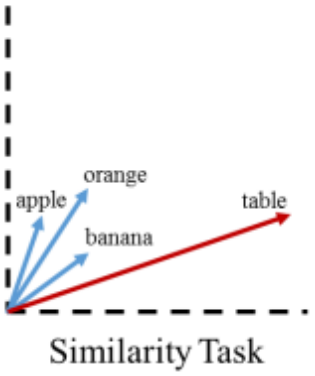

FIGURE 3 – représentation graphique de la tâche de similarité.

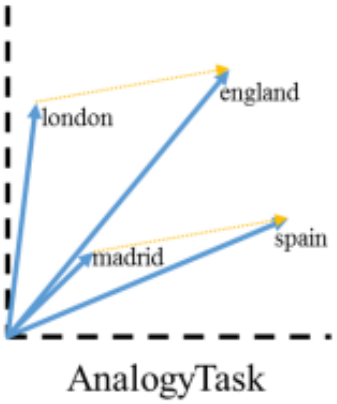

FIGURE 4 – représentation graphique de la tâche d'analogie.

4.2 Tâche d'analogie

La tâche d'analogie proposée par [11] nous permet d'évaluer la similitude relationnelle entre les paires de mots. Cette tâche consiste à répondre à des questions d'analogie basées sur les similitudes cosinus calculées sur les plongements de mots. La relation de similitude entre des paires de mots du même type (e.g. genre) peut être transformée en une question d'analogie., Lorsqu'on leur donne une paire de mots a et a et un troisième mot b, la relation d'analogie entre a et a peut être utilisé pour trouver le mot correspondant b à b. Mathématiquement, il est exprimé comme :

$$a- > a^* :: b- > \_\_$$

où le blanc (__) est b^*. Un exemple pourrait être

$$homme- > roi^* :: femme- > reine^*$$

La méthode 3CosAdd [12] résout pour b^* en utilisant l'équation suivante :

$$b^* = argmax(cos(b', a^* - a + b))$$

Ainsi, une similitude cosinus élevée signifie que les vecteurs partagent une direction similaire. Cependant, il est important de noter que la méthode 3CosAdd normalise les longueurs des vecteurs en utilisant la similarité cosinus [12]. Alternativement, il existe la méthode 3CosMul [13], qui est définie comme :

$$b^* = argmax\frac{cos(b', b)cos(b', a^*)}{cos(b', a^*) + \epsilon}$$

où $\epsilon = 0,001$ est utilisé pour empêcher la division par zéro. La méthode 3CosMul a le même effet en prenant le logarithme de chaque terme avant la sommation. En d'autres termes, les petites

différences sont agrandies tandis que les grandes sont supprimées. On voit donc que la méthode 3CosMul offre un meilleur équilibre sous différents aspects.

	bruni_men	luong_rare	radinsky_mturk	ws353	ws353_relatedness	ws353_similarity
PMI	-0.016	-0.069	0.085	-0.059	0.108	-0.147
$NPMI$	0.157	-0.049	0.576	0.493	0.521776	0.523679
PMI^2	0.175	-0.046	0.583	0.545	0.579	0.564
MPI^3	0.18	-0.046	0.59	0.548	0.578	0.567
$PPMI$	0.157	-0.049	0.576	0.494	0.521	0.523
$SVDd = 50$	0.118	-0.0	0.608	0.527	0.516	0.569
$SVDd = 100$	0.142	0.011	**0.627**	0.576	0.58	0.586
$SVDd = 300$	0.185	0.039	0.583	0.61	0.63	0.623
$SVDd = 500$	0.203	0.04	0.588	0.621	**0.637**	0.621
$SVDd = 700$	**0.21**	0.06	0.564	**0.622**	0.635	**0.634**
$SVDd = 900$	0.203	**0.061**	0.556	0.618	0.632	0.621

TABLE 1 – Résultats de la tâche de similarité de différentes méthodes de transformation.

	EN-TOM-ICLR13-SEM	EN-TOM-ICLR13-SYN	google	msr
PMI	0.313	0.208	0.256	0.18
$NPMI$	**0.395**	0.282	**0.333**	0.212
PMI^2	0.388	0.233	0.303	0.18
MPI^3	0.331	0.158	0.236	0.143
$PPMI$	0.394	0.279	0.331	0.203
$SVDd = 50$	0.108	0.195	0.155	0.166
$SVDd = 100$	0.145	0.271	0.214	0.216
$SVDd = 300$	0.266	**0.308**	0.289	**0.243**
$SVDd = 500$	0.316	0.279	0.295	0.223
$SVDd = 700$	0.317	0.266	0.29	0.205
$SVDd = 900$	0.311	0.263	0.284	0.201

TABLE 2 – Résultats de la tâche d'analogie 3cosAdd[12] de différentes méthodes de transformation.

	EN-TOM-ICLR13-SEM	EN-TOM-ICLR13-SYN	google	msr
PMI	0.006	0.022	0.014	0.014
$NPMI$	0.318	0.297	0.306	0.213
PMI^2	**0.39**	0.261	**0.32**	0.188
MPI^3	0.368	0.198	0.275	0.134
$PPMI$	0.344	0.298	0.319	0.215
$SVDd = 50$	0.099	0.156	0.13	0.14
$SVDd = 100$	0.139	0.251	0.2	0.208
$SVDd = 300$	0.28	**0.331**	0.308	**0.258**
$SVDd = 500$	0.331	0.303	0.316	0.23
$SVDd = 700$	0.342	0.293	0.315	0.212
$SVDd = 900$	0.341	0.285	0.31	0.208

TABLE 3 – Résultats de la tâche d'analogie 3cosMul[13] de différentes méthodes de transformation.

5 Conclusion

Nous avons proposé des méthodes de transformation basées sur le PMI, Un package permettant de réaliser cette étude est développé en python3.

REFERENCES

[1] Yaghoobzadeh, Y. ; Schütze, H. : Intrinsic subspace evaluation of word embedding representations. in Proc. of the 54th Annual Meeting of the Association for Computational Linguistics, vol. 1, 2016, 236–246.

[2] Qiu, L. ; Cao, Y. ; Nie, Z. ; Yu, Y. ; Rui, Y. : Learning word representation considering proximity and am

[3] Hellrich, J. ; Hahn, U. : Don't get fooled by word embeddings : better watch their neighborhood. in Digital Humanities 2017–Conf. Abstracts of the 2017 Conf. of the Alliance of Digital Humanities Organizations (ADHO). Montréal, Quebec, Canada, 2017, 250–252.

[4] Gladkova, A. ; Drozd, A. : Intrinsic evaluations of word embeddings : what can we do better ? in Proc. of the 1st Workshop on Evaluating Vector-Space Representations for NLP, 2016, 36–42. [32] Mu, J. ; Bhat, S. ; Viswanath, P. : All-but-the-top : simpl

[5] Mu, J. ; Bhat, S. ; Viswanath, P. : All-but-the-top : simple and effective postprocessing for word representations. arXiv preprint arXiv :1702. 01417, 2017.

[6] Wang, B. ; Chen, F. ; Wang, A. ; Kuo, C.-C.J. : Post-processing of word representations via variance normalization and dynamic embedding. in 2019

[7] P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov, Enriching Word Vectors with Subword Information

[8] Jeffrey Pennington, Richard Socher, Christopher Manning, Global Vectors for Word Representation

[9] Franc¸ois Role and Mohamed Nadif, HANDLING THE IMPACT OF LOW FREQUENCY EVENTS ON CO-OCCURRENCE BASED MEASURES OF WORD SIMILARITY A Case Study of Pointwise Mutual Information

[10] [Levy et al., 2014] Levy, O., Goldberg, Y. et Ramat-Gan, I. (2014). Linguistic regularities in sparse and explicit word representations. In CoNLL, pages 171–180.

[11] [Mikolov et al., 2013c] Mikolov, T., Yih, W.-t. et Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In HLT-NAACL, pages 746–751.

[12] Mikolov, T. ; Yih, W.-t. ; Zweig, G. : Linguistic regularities in continuous space word representations. in Proc. of the 2013 Conf. of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, 2013, 746–75 1.

[13]Levy, O. ; Goldberg, Y. : Linguistic regularities in sparse and explicit word representations. in Proc. of the Eighteenth Conf. on Computational Natural Language Learning, 2014, 171–180