# Plongement de métrique pour le calcul de similarité sémantique à l'échelle

Julien Subercaze*, Christophe Gravier*, Frédérique Laforest*

* Université de Lyon, F-42023, Saint-Etienne, France
CNRS, UMR5516, Laboratoire Hubert Curien, F-42000, Saint-Etienne, France,
Université de Saint-Etienne, Jean Monnet, F-42000, Saint-Etienne, France

**Résumé.** Nous explorons le plongement de la métrique de plus court chemin dans l'hypercube de Hamming, dans l'objectif d'améliorer les performances de similarité sémantique dans Wordnet (Subercaze et al. (2015)). Nous montrons que bien qu'un plongement isométrique est impossible en pratique, nous obtenons de très bons plongements non isométriques. Nous obtenons une amélioration des performances de trois ordres de grandeur pour le calcul de la similarité de Leacock et Chodorow (LCH).

## 1 Introduction

Le concept de similarité sémantique encode la distance conceptuelle entre deux unités de langage. Quand les mots sont les unités de discours, cette similarité est au cœur de nombreuses tâches de TALN. Elle est notamment essentielle pour la désambiguisation (Basile et al. (2014)). Deux approches dominent la mise en œuvre du calcul de la similarité sémantique.

Les approches basées sur les bases de connaissance exploitent aussi bien la structure de la taxonomie Leacock et Chodorow (1998), que son contenu Banerjee et Pedersen (2002) ou les deux Jiang et Conrath (1997). Ces approches ignorent les informations contextuelles et utilisent des bases décrites manuellement. A l'opposé, la sémantique statistique encode la similarité sémantique en se basant sur des observations statistiques, par exemple sur les occurrences dans un corpus. Cependant cette approche est largement limité quant au volume des données qu'elle peut traiter. Les nouvelles approches de plongement de sémantique statistique dans des espaces vectoriels compact (*word embedding*, Collobert et Weston (2008)) apportent une réponse efficace à ce problème. Les architectures neuronales permettent un traitement de larges volumes de données au prix d'un temps d'entrainement de l'ordre de plusieurs jours. La popularité des approches neuronales montre un enthousiasme certain pour des approches efficaces de calcul de similarité entre des paires de mots.

Dans cet article, nous proposons un plongement de la similarité sémantique de Leacock et Chodorow (1998) dans un hypercube de Hamming de dimensions alignées sur la taille des mots processeurs. La similarité de Leacock et Chodorow, basée sur la métrique de plus court chemin dans la relation d'hypernymie de Wordnet, est une des mesures les plus précises. Dans l'évaluation de Miller et Charles (1991) elle atteint le deuxième rang. Ses performances sont légèrement dépassées par l'approche de Jiang et Conrath (1997), basée sur le contenu.

Les techniques de plongement dans l'hypercube de Hamming permette de bénéficier de temps de calcul très performants sur les processeurs modernes. Nous montrons tout d'abord qu'un plongement isométrique, bien que possible, requiert un nombre de dimensions trop élevé pour être utilisable dans la pratique. Notre résultat principal est la construction d'un plongement non isométrique de la métrique de plus court chemin d'arbre, tout en maintenant une forte corrélation avec les distances originales ($r = .819, \rho = .829$). Nos expériences montrent un gain de trois ordres de grandeur en temps de calcul par rapport à l'implémentation de référence.

## 2 Plongement de la métrique du plus court chemin

Nous introduisons tout d'abord les notations. Soit $H_2^n$ un hypercube de n dimensions dont les nœuds sont labellisés par les $2^n$ n-tuples binaires. Deux nœuds sont adjacents si et seulement si leurs n-tuples correspondants diffèrent en exactement une position, c.a.d leur distance de Hamming($\ell_1$) est égale à un. Dans la suite de l'article, $Q^n$ dénote l'espace métrique composé de $H_2^n$ doté de ($\ell_1$).

Notre problème est le suivant : définir une fonction $f$ qui, à chaque nœud $w$ de la taxonomie (Wordnet pour LCH), associe un nœud dans $Q^n$ tel que, : $\forall(w_i, w_j), d(w_i, w_j) = \lambda \cdot \ell_1(f(w_i), f(w_j))$, avec $\lambda$ un scalaire. Pour des raisons de faisabilité, la complexité de cette fonction doit être *raisonnable* et $n$ le plus petit possible.

**Limites théoriques** La relation d'hypernymie de Wordnet forme un treillis. Une première approche serait de réaliser un plongement isométrique du treillis doté de sa métrique de plus court chemin dans $Q^n$. Cependant, les travaux de Deza et Laurent (1997) ont montré qu'un tel plongement existe mais requiert $2^k$ dimensions, avec $k$ le nombre de nœuds dans le treillis. De telles dimensions ne permettent pas de plonger un treillis de plus de 60 nœuds. La hiérarchie de Wordnet compte plusieurs dizaines de milliers de nœuds. Cette solution est donc impossible à mettre en œuvre. La borne de Deza n'est pas stricte, mais il faudrait une amélioration plus que drastique pour obtenir un plongement d'un quelconque intérêt pratique.

**Plongement d'arbre** Pour réduire la dimension du plongement, nous entreprenons de simplifier la structure du treillis en un arbre. Le treillis de Wordnet est en effet peu dense, nous obtenons un arbre en coupant 1,300 liens, soit moins d'un pourcent des liens originaux. La nature de la coupe fera l'objet de recherche plus poussées ; dans cet article, nous suivons une heuristique simple : pour un nœud ayant plusieurs hypernymes, nous préservons le premier dans l'ordre de Wordnet. Nos expériences montrent que la similarité de LCH sur l'arbre obtenu a de fortes corrélations avec la similarité calculée sur le treillis ($r = .919, \rho = .931$).

Wilkeit (1990) a montré que tout arbre k-aire de taille n peut être plongé dans $Q^{n-1}$. Nous donnons ici un algorithme en temps linéaire ($O(n^5)$ pour Wilkeit dans un cas plus général), qui servira de base à la construction de notre solution. Dans un parcours pré-ordre en profondeur, si le nœud a $k$ fils, on assigne le n-tuple suivant au $i$-ème fils : on concatène $k$ zéros au n-tuple du parent et on met le $i$-ème bit à 1. Le résultat obtenu est un plongement nécessitant plusieurs dizaines de milliers de dimensions, loin de notre objectif de plusieurs mots processeurs.

Le grand nombre de dimensions est une conséquence de cette concaténation de signature, qui résulte en une faible densité de bits mis à un. A l'opposé, l'approche de Chen et Stallmann (1995) utilise un code de Gray. Cette méthode est optimale en terme de dimensions, mais introduit une erreur qui rend ce plongement inutilisable. Le tableau 1 montre la compacité du plongement (17 bits) mais au prix de très faibles corrélations ($r = .235$ and $\rho = .186$).

Une recherche exhaustive du meilleur plongement pour une dimensions cible arbitraire est hors de portée au vue des puissances de calcul actuelles. Pour un plongement de dimension $n$ et un arbre de $r$ nœuds, l'espace de recherche contient $C = \frac{(2^n)!}{(n-r)!}$ possibilités. Pour $n = 17$ et $r = 87{,}000$, on obtient $C > 10^{10,000}$. Pour un alignement sur un mot processeur ($n = 64$), on a $C > 10^{100,000}$.

# 3 Plongement non-isométrique

Notre approche est un compromis entre le plongement isométrique et l'approche de Chen et Stallmann (1995) basée sur les codes de Gray. Notre plongement introduisant une erreur, nous pouvons choisir de préserver une distance d'arbre plus qu'une autre, c'est-à-dire soit la distance parent-enfants, soit la distance inter-enfants.

Par conséquence, la conception de la solution est basée sur les propriétés de l'arbre qui sera plongé dans l'hypercube. Nous analysons les caractéristiques de l'arbre issu de la coupe dans le treillis. L'arbre possède un facteur de branchement de 4.9 avec un écart-type de 14. 96% des nœuds ont un facteur de branchement inférieur à 20. A l'inverse, la profondeur de l'arbre est stable avec une moyenne de 8.5, un écart-type de 2 et un maximum de 18. Au vu de la stabilité de la profondeur, nous décidons de favoriser la distance parent-enfant. Afin de minimiser les dimensions du plongement, nous souhaitons allouer moins de $k$ bits pour un nœud possédant $k$ fils. Notre approche se base sur les principes suivants :

**Héritage de branche :** chaque nœud hérite de la signature de son père, mais à la différence du plongement isométrique, l'extension n'induit pas une extension globale sur tous les nœuds. Cette approche permet de garantir une certaine compacité de la structure.

**Préservation de la distance parent enfant :** en allouant moins de bits que le nombre requis pour un plongement isométrique, notre approche introduit naturellement une erreur. Nous faisons le choix d'allouer, pour un noeud avec $k$ fils, $\lceil log_2(k+1) \rceil$ pour étendre cette signature, afin de garantir l'unicité des signatures.

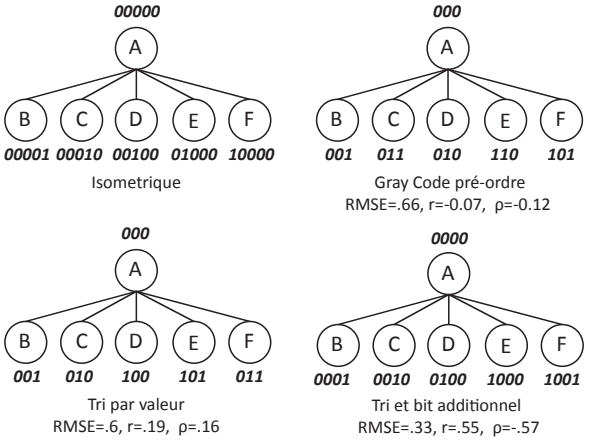


FIG. 1 – *Approches de réduction de dimensions.*

**Alignement sur les mots processeurs :** Les deux principes précédemment énoncés permettent d'obtenir un plongement compact pour les arbres de faible profondeur, comme celui obtenu à partir de l'hypernymie de Wordnet. Cependant la dimension du plongement obtenu n'est pas nécessairement aligné sur la taille d'un mot CPU $W$ : $kW \leq D \leq (k+1)W$. Nous souhaitons alors exploiter les $(k+1)W - D$ bits inutilisés mais qui seront de toutes façons manipulés par le processeur. Pour cela, nous proposons de trier

les nœuds selon une valeur $v(i), i \in N$, de façon à déterminer quels nœuds sont éligibles à l'obtention d'un ou plusieurs bits supplémentaires. Puisque notre approche favorise la conservation de la distance père-fils, nous souhaitons favoriser l'allocation pour les nœuds proches de la racine et qui sont les parents d'une vaste descendance. La formule suivante donne un compromis entre ces deux critères : $v(i) = (max_{depth} - depth(i)) \cdot log(size_{branch}(i))$.

Dans le but d'améliorer la qualité du plongement, nous introduisons deux optimisations supplémentaires et optionnelles. La première est le tri par fils : Nous choisissons de préserver au mieux les fils qui ont de nombreux descendants. La seconde est le tri par valeur, présenté en figure 3. Parmi les $2^{\lceil log_2(k+1) \rceil}$ signatures possibles, seulement $k+1$ sont utilisées dans notre approche. Par exemple dans le cas de cinq fils (figure 3), nous allouons trois bits pour six signatures. Nous favorisons l'utilisations de signatures qui introduisent une erreur minimale, à la différence de l'approche basée sur les codes de Gray.

# 4 Expérimentations

Dans cette section, nous mettons en place deux expériences afin d'évaluer la qualité et les performances de notre approche. Dans la première expérience, nous testons la qualité du plongement en mesurant la corrélation avec la distance originale LCH ainsi qu'avec la distance LCH calculée sur l'arbre obtenu par coupe.

La deuxième expérience compare les performances d'exécution de notre approche avec l'implémentation de référence qui intègre notamment un mécanisme de cache pour réduire ses temps de calcul.

Notre algorithme, `FSE` pour *Fast Similarity Embedding* est implémenté en Java et disponible publiquement à l'adresse suivante `http://bit.ly/FSELCH`.

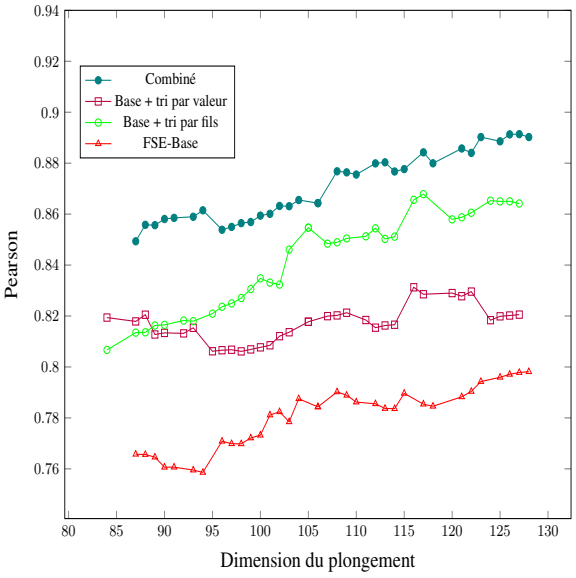


FIG. 2 – *FSE : influence des optimisations et des dimensions sur la corrélation avec la distance LCH calculée sur l'arbre de Wordnet.*

Nous implémentons différentes versions de l'algorithme. `FSE-Base` est l'algorithme de base, ne contenant aucune des deux optimisations présentées à la section précédente. Deux autres versions permettent l'utilisation respective des optimisations, la version `FSE-Best` contenant ces deux optimisations.

## 4.1 Qualité du plongement

Tout d'abord nous mesurons la corrélation des distances obtenus par plongement avec les distances obtenues sur l'arbre, afin de mesurer l'impact des optimisations. La figure 2 montre l'infuence des dimensions et des optimisations sur la valeur du coefficient de corrélation de Pearson. La version de base obtient sa meilleure valeur $r = .77$ pour une dimension de 128 bits. En ce qui concerne les optimisations, le tri par fils est plus efficace que le tri par valeur pour toutes les dimensions au delà de 90. Finalement, la combinaison des optimisations permet d'obtenir une bien meilleure corrélation ($r = .89$) que les autres versions de l'algorithme.

| Plongement | Bits | Pearson | Spearman |
|---|---|---|---|
| Chen et al. | 17 | .235 | .186 |
| FSE-Base | 84 | .699 | .707 |
| **FSE-Best** | **128** | **.819** | **.829** |
| Isometric | 84K | .919 | .931 |

TAB. 1 – *Corrélations entre LCH, le plongement isométrique et FSE pour toutes les distances sur les paires de mots de Wordnet-Core (p-values* $\leq 10^{-14}$*).*

Nous mesurons ensuite la corrélation avec la mesure LCH sur le treillis originial. Nous calculons la corrélation sur cinq millions de distances depuis le jeu de données *Wordnet-Core* [1]. La corrélation entre la mesure LCH obtenue sur l'arbre et sur le treillis donne la borne théorique que nous pouvons atteindre ($r = .919$ et $\rho = .931$). FSE-Best présente ici des corrélations particulièrement intéressante, avec $r = .819$ et $\rho = .829$, relativement proches de la borne théorique. Notre approche requiert 650 fois moins de bits que le plongement isométrique, tout en maintenant une très forte corrélation avec la mesure de similarité LCH.

## 4.2 Temps d'exécution

Le tableau 4.2 présente les temps d'exécution pour la similarité LCH. Nous comparons nos résultats avec les temps d'exécution de la librairie WS4J [2], une librairie efficace utilisant un cache en mémoire centrale.

| Algo | Measure | Nombre de pairs ($n$) | | | | |
|---|---|---|---|---|---|---|
| | | $10^3$ | $10^4$ | $10^5$ | $10^6$ | $10^7$ |
| WS4J | $10^3 \cdot$ ms | 0.156 | 1.196 | 11.32 | 123.89 | 1,129.3 |
| FSE-Best | ms | 0.04 | 0.59 | 14.15 | 150.58 | 1,482 |
| **speedup** | | ×3900 | ×2027 | ×800 | ×822 | ×762 |

TAB. 2 – *Temps d'exécution en millisecondes pour le calcul de similarité mot à mot.*

Le calcul de la distance de Hamming nécessitant peu d'instruction processeurs (XOR et POPCNT), FSE offre sans surprise de bien meilleures performances que WS4J, jusqu'à trois ordres de grandeur, dans le cas d'un faible nombre de paires.

Ce facteur descend à 800 fois pour de très grands volumes de données, ceci est notamment dû au mécanisme de cache de WS4J qui stocke des distances intermédiaires de plus court chemin et gagne en efficacité avec le nombre de requêtes.

1. https://wordnet.princeton.edu/wordnet/download/standoff/
2. https://code.google.com/p/ws4j/

# 5 Conclusion

Dans cet article, nous montrons une nouvelle approche de plongement de métrique de plus court chemin dans un arbre pour améliorer substantiellement les performances de calcul de la similarité de Leacock et Chodorow. Nous montrons que le plongement isométrique du treillis de la hiérarchie de Wordnet est faisable mais n'est d'aucun intérêt en pratique. Nous effectuons une coupe dans le treillis pour obtenir un arbre puis nous avons proposé une heuristique pour plonger l'arbre muni de sa distance du plus court chemin dans l'hypercube de Hamming dans des dimensions alignées sur celles des mots processeurs. Nos expérimentations montrent une amélioration de deux à trois ordres de grandeur tout en maintenant une très forte corrélation.

# Références

Banerjee, S. et T. Pedersen (2002). An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing*, pp. 136–145.

Basile, P., A. Caputo, et G. Semeraro (2014). An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In *Proc. of COLING*, pp. 1591–1600.

Chen, W.-K. et M. F. Stallmann (1995). On embedding binary trees into hypercubes. *Journal of Parallel and Distributed Computing 24*(2), 132–138.

Collobert, R. et J. Weston (2008). A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proc. of the 25th international conference on Machine learning*, pp. 160–167. ACM.

Deza, M. et M. Laurent (1997). *Geometry of Cuts and Metrics*. Springer, 588 pages.

Jiang, J. J. et D. W. Conrath (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *Proc. of CICLing*.

Leacock, C. et M. Chodorow (1998). Combining local context and wordnet similarity for word sense identification. *WordNet : An electronic lexical database 49*(2), 265–283.

Miller, G. A. et W. G. Charles (1991). Contextual correlates of semantic similarity. *Language and cognitive processes 6*(1), 1–28.

Subercaze, J., C. Gravier, et F. Laforest (2015). On metric embedding for boosting semantic similarity computations. In *Association of Computational Linguistics*.

Wilkeit, E. (1990). Isometric embeddings in hamming graphs. *Journal of Combinatorial Theory, Series B 50*(2), 179–197.

# Summary

In this paper, we explore the embedding of the shortest-path metrics from a knowledge base (Wordnet) into the Hamming hypercube, in order to enhance the computation performance. We show that, although an isometric embedding is untractable, it is possible to achieve good non-isometric embeddings. We report a speedup of three orders of magnitude for the task of computing Leacock and Chodorow (LCH) similarities while keeping strong correlations ($r = .819, \rho = .826$).