Représentation vectorielle de sens pour la désambiguïsation lexicale à base de connaissances

Loïc Vial, Benjamin Lecouteux, Didier Schwab

Représentation vectorielle de sens pour la désambiguïsation lexicale à base de connaissances

Loïc Vial, Benjamin Lecouteux, Didier Schwab
GETALP – LIG – Univ. Grenoble Alpes
{loic.vial, benjamin.lecouteux, didier.schwab}
@univ-grenoble-alpes.fr

RÉSUMÉ

Dans cet article, nous proposons une nouvelle méthode pour représenter sous forme vectorielle les sens d'un dictionnaire. Nous utilisons les termes employés dans leur définition en les projetant dans un espace vectoriel, puis en additionnant les vecteurs résultants, avec des pondérations dépendantes de leur partie du discours et de leur fréquence. Le vecteur de sens résultant est alors utilisé pour trouver des sens reliés, permettant de créer un réseau lexical de manière automatique. Le réseau obtenu est ensuite évalué par rapport au réseau lexical de WordNet, construit manuellement. Pour cela nous comparons l'impact des différents réseaux sur un système de désambiguïsation lexicale basé sur la mesure de Lesk. L'avantage de notre méthode est qu'elle peut être appliquée à n'importe quelle langue ne possédant pas un réseau lexical comme celui de WordNet. Les résultats montrent que notre réseau automatiquement généré permet d'améliorer le score du système de base, atteignant quasiment la qualité du réseau de WordNet.

ABSTRACT

Sense Embeddings in Knowledge-Based Word Sense Disambiguation

In this paper, we develop a new way of creating sense vectors for any dictionary, by using an existing word embeddings model, and summing the vectors of the terms inside a sense's definition, weighted in function of their part of speech and their frequency. These vectors are then used for finding the closest senses to any other sense, thus creating a semantic network of related concepts, automatically generated. This network is hence evaluated against the existing semantic network found in WordNet, by comparing its contribution to a knowledge-based method for word sense disambiguation. This method can be applied to any other language which lacks such semantic network, as the creation of word vectors is totally unsupervised, and the creation of sense vectors only needs a traditional dictionary. The results show that our generated semantic network improves greatly the WSD system, almost as much as the manually created one.

MOTS-CLÉS : Représentation vectorielle de sens, Désambiguïsation Lexicale.

KEYWORDS: Sense Embeddings, Word Sense Disambiguation.

1 Introduction

Dans le Traitement Automatique des Langues (TAL), la Désambiguïsation Lexicale (DL) consiste à attribuer le sens le plus probable à un mot donné dans un document, à partir d'un inventaire prédéfini de sens. Les méthodes état de l'art sont souvent basées sur des approches supervisées, comme l'évoque Navigli (2009), où les modèles sont appris sur de grands corpus de mots annotés. Les modèles appris sont alors utilisés pour annoter les mots avec un sens dans un contexte donné. *A contrario*, les méthodes non supervisées ont l'avantage de nécessiter beaucoup moins de ressources pour fonctionner : elles n'utilisent pas de corpus préalablement annoté pour l'apprentissage. Il en découle en général une meilleure couverture sur les données de test, car elles sont moins tributaires des sens observés lors de l'apprentissage. Un autre avantage des méthodes non supervisées est qu'elles sont beaucoup plus généralisables et ne sont pas limitées à des langues très dotées telles que l'anglais. Cet aspect est important à relever, car l'annotation de corpus en sens est extrêmement lourde : peu de langues disposent de ce type de ressource. [1]

Pour ces raisons, nos travaux s'articulent autour des méthodes non supervisées et présentent une approche originale améliorant les performances existantes, basée sur des vecteurs de sens.

Les méthodes récentes et état de l'art de représentation vectorielle de mots, telles que Mikolov *et al.* (2013) (Word2Vec), Pennington *et al.* (2014) (GloVe) ou Levy & Goldberg (2014) (vecteurs basés sur des interdépendances), ont montré des gains intéressants dans de nombreuses tâches du TAL telles que la traduction automatique, le calcul de similarité entre mots ou encore en désambiguïsation lexicale, où ils ont été employés dans des méthodes récentes telles que Chen *et al.* (2014) ou Yuan *et al.* (2016).

Dans cet article nous présentons une méthode de représentation vectorielle pour des sens : où des vecteurs vont représenter les sens présents dans la base lexicale *Princeton WordNet* (Miller, 1995). La méthodologie utilisée pour créer ces vecteurs est décrite dans la section 2. Ces vecteurs de sens sont ensuite évalués en comparant leurs performances en tant que réseau lexical appliqué à un système de DL à base de connaissances dans la section 3.1.

Ces vecteurs de sens sont utilisés pour étendre les définitions des sens présents dans *WordNet* en concaténant les définitions des sens proches, d'une manière similaire au Lesk étendu de Banerjee & Pedersen (2002), sauf qu'au lieu de considérer que deux sens sont proches car ils partagent une relation syntaxique ou sémantique dans le réseau *WordNet*, nous considérons que deux sens sont proches en fonction de leur similarité cosinus. Dans la partie relative aux expérimentations nous explorons comment exploiter au mieux cette similarité.

2 Création de vecteurs de sens

Notre modèle pour représenter sous forme vectorielle les sens d'un dictionnaire repose sur leur définition, ainsi qu'un modèle pré-existant de vecteurs de mots. Dans la pratique, la méthode est relativement similaire à celle présentée par Ferrero *et al.* (2017) qui crée des vecteurs de phrases pour la détection de plagiat inter-lingue. Nos vecteurs de sens sont calculés comme la somme normée de tous les vecteurs des termes présents dans la définition du sens donné. Ces vecteurs sont pondérés

1. Moins de dix langues possèdent au moins un corpus annoté en sens via WordNet, listées sur `http://globalwordnet.org/wordnet-annotated-corpora/`, en date du 16 avril 2017.

en fonction de leur partie du discours (*Part Of Speech*, ou POS) : nom, verbe, adjectif ou adverbe, et également pondérés par leur *Inverse Document Frequency* (ou IDF) : l'inverse de leur nombre d'occurrence dans le dictionnaire entier. Plus formellement nous avons :

— $D(S) = \{w_0, w_1, w_2, \ldots, w_n\}$ la définition du sens S dans le dictionnaire
— $pos(w_n) = \{n, v, a, r\}$ la partie du discours du terme w_n (nom, verbe, adjectif, ou adverbe)
— $weight(pos)$ le poids associé à une partie du discours
— $idf(w_n)$ la valeur IDF de w_n

La définition du vecteur du sens S, notée $\phi(S)$ correspond à :

$$\phi(S) = \sum_{i=0}^{n} (\phi(w_n) \times weight(pos(w_n)) \times idf(w_n))$$

$\phi(S)$ est ensuite normé afin d'avoir la même taille que les vecteurs de mots (qui sont eux aussi normés). Les poids attribués aux POS sont les même que ceux utilisés par Ferrero *et al.* (2017).

Nous avons créé cinq modèles de vecteurs de sens, tous basés sur le vocabulaire et les définitions de *WordNet* 3.0, mais exploitant des modèles de vecteurs de mots différents. Les cinq modèles utilisés sont :

1. Un modèle pré-entraîné et proposés par Mikolov *et al.* (2013) [2]. Ce modèle a été entraîné sur environ 100 milliards de mots issus du corpus de nouvelles de Google. La taille du vocabulaire est d'environ de 3 millions de mots et les vecteurs ont une dimension de 300
2. Un modèle de Pennington *et al.* (2014)'s GloVe [3], entraîné sur 42 milliards de mots issus de *Common Crawl*. Le vocabulaire est de 2 millions de mots et les vecteurs ont une taille de 300.
3. Un modèle de Levy & Goldberg (2014) [4]. L'apprentissage a été effectué sur Wikipedia, le vocabulaire est de 175 000 mots et la taille des vecteurs 300.
4. Le meilleur des modèles de prédiction de Baroni *et al.* (2014) [5]. La taille du vocabulaire est de 300 000 et la taille des vecteurs est de 400.
5. Et finalement, le meilleur modèle à base de comptage également créé par Baroni *et al.* (2014)[5] de dimension 500 et une taille de vocabulaire identique au précédent modèle.

Dans la section expérimentale, nous comparons les performances de chacun de ces modèles dans une tâche de désambiguïsation lexicale en anglais. *WordNet* 3.0 est composé de 206 941 sens, et nous avons donc créé un vecteur pour chacun d'eux. Tous nos modèles de vecteurs de sens sont rendus publics sur la page de l'article, afin de reproduire les expériences, à l'adresse suivante : `https://github.com/getalp/WSD-TALN2017-Vialetal`.

3 Intégration à un système de désambiguïsation lexicale

Les vecteurs de sens ainsi générés sont évalués sur une tâche de désambiguïsation lexicale (DL). L'idée est d'exploiter le modèle en lieu et place d'un réseau lexical. Pour cela nous comparons les

2. `https://code.google.com/archive/p/word2vec/`
3. `https://nlp.stanford.edu/projects/glove/`
4. `https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/`
5. `http://clic.cimec.unitn.it/composes/semantic-vectors.html`

sens entre eux via la distance cosinus, et selon un seuil δ déterminé empiriquement, les sens les plus proches sont sélectionnés. Les sens sélectionnés vont alors contribuer à l'algorithme de DL au niveau de sa mesure de similarité locale, dans un mode de fonctionnement similaire à celui présenté par Banerjee & Pedersen (2002) à la différence qu'eux exploitent le réseau lexical de *WordNet*.

3.1 Algorithme de désambiguisation

Le système de DL que nous proposons est composé de deux éléments : un algorithme local qui calcule un score de similarité pour une paire de sens, et un algorithme global qui va chercher la meilleure combinaison de sens à l'échelle du document, en utilisant l'algorithme local.

L'algorithme global est basé sur une heuristique permettant de ne pas explorer exhaustivement l'ensemble des combinaisons possibles, ce qui engendrerait des coûts calculatoires trop élevés.

L'heuristique utilisée dans notre système exploite un algorithme de recherche à base de coucous, comme celui présenté dans Vial *et al.* (2017). Les auteurs montrent que cet algorithme global est actuellement un des plus efficace connu pour la tâche de désambiguïsation.

Notre implémentation réutilise les meilleurs paramètres présentés dans Vial *et al.* (2017) (un seul coucou, le *Levy location* à 5 et le *Levy scale* à 0,5).

Dans l'ensemble de nos expériences nous avons utilisé un grand nombre d'itérations (300 000) afin de réduire l'effet aléatoire d'une expérience à une autre. De plus, nous exécutons en moyenne une dizaine d'exécutions pour en extraire un score moyen. De cette manière nous assurons un écart-type inférieur à 0,1 : les résultats sont ainsi stables et reproductibles.

L'algorithme local est l'élément central de notre système et c'est pour l'amélioration de ce dernier qu'est utilisé le réseau lexical. Comme système étalon nous utilisons une mesure de Lesk standard. Cet algorithme retourne, comme mesure de similarité, le nombre de mots communs à deux définitions de sens. Formellement, si nous notons $D(S) = \{w_1, w_2, \ldots, w_n\}$ la définition de S, alors la mesure de Lesk entre deux sens S_1 et S_2 notée $Lesk(S_1, S_2)$ est la suivante :

$$Lesk(S_1, S_2) = |D(S_1) \cap D(S_2)|$$

Comme second système étalon et dans l'objectif de mesurer la qualité de notre réseau lexical généré automatiquement, nous utilisons la mesure de Lesk étendue de Banerjee & Pedersen (2002).

Cette mesure considère à la fois la définition du sens cible mais également les définitions de tous les sens qui ont une relation avec le sens cible (hyponyme, etc.). Si l'on note $rel(S)$ l'ensemble des sens reliés à S à travers un lien explicite dans *WordNet*, alors la mesure de Lesk étendue entre les sens S_1 et S_2 notée $ExtLesk(S_1, S_2)$ est la suivante :

$$ExtLesk(S_1, S_2) = \left|(D(S_1) \cup_{r \in rel(S_1)} D(r)) \cap (D(S_2) \cup_{r \in rel(S_2)} D(r))\right|$$

3.2 Extension de définitions grâce aux vecteurs de sens

Notre méthode utilisant les vecteurs de sens décrits dans la section 2 est d'étendre l'algorithme de Lesk de la même manière que Banerjee & Pedersen (2002) qui exploite les relations entre les sens, lors du calcul de la similarité entre deux sens.

Étant donné que nos vecteurs de sens sont créés à partir des définitions d'un dictionnaire, les termes utilisés dans ces définitions sont très importants. Pour cette raison, quand nous voulons capturer les sens proches d'un sens S, nous ne prenons pas en compte automatiquement tous les sens qui ont une forte similarité cosinus avec S. En effet, si nous faisions cela, nous considérerions aveuglément comme "proche" tous les sens dont les définitions sont formées de mots similaires à ceux la définition de S. Ainsi nous filtrons également les sens proches du lemme du sens S, dans l'espace des vecteurs de mots. Ce filtrage assure de capturer des sens proches de l'*idée* du lemme capturée par le modèle de vecteurs de mots.

Dans notre approche nous avons eu la volonté de pouvoir effectuer des similarités entre vecteurs de sens et vecteurs de mots. Ceci est rendu possible par le fait que les vecteurs de sens sont une somme de vecteurs de mots, et donc sont de même dimension.

Si l'on note $rel(S, \delta_1, \delta_2)$ l'ensemble des sens liés à S. δ_1 est le seuil de sélection sur la similarité cosinus avec le vecteur du lemme, δ_2 est un seuil de sélection sur la similarité avec le vecteur du sens. La fonction $rel(S, \delta_1, \delta_2)$ est calculée de la manière suivante :

$$rel(S_1, \delta_1, \delta_2) = \{S_2 \in Senses \mid cosine(\phi(lemma(S_1)), \phi(S_2)) > \delta_1, cosine(\phi(S_1), \phi(S_2)) > \delta_2\}$$

Finalement, le calcul pour notre nouvel algorithme local de DL que nous appellons $VecLesk(S_1, S_2, \delta_1, \delta_2)$, peut être décrit de la manière suivante :

$$VecLesk(S_1, S_2, \delta_1, \delta_2) = \left|(D(S_1) \cup_{r \in rel(S_1, \delta_1, \delta_2)} D(r)) \cap (D(S_2) \cup_{r \in rel(S_2, \delta_2, \delta_2)} D(r))\right|$$

Dans la section suivante nous évaluons ce nouvel algorithme local sur deux tâches de DL en le comparant avec deux systèmes étalons : l'algorithme de Lesk classique, et de Lesk étendu.

4 Évaluation

Afin d'expérimenter notre nouveau modèle de vecteurs de sens, nous l'exploitons pour effectuer une extension lexicale de définitions de dictionnaire avec pour objectif d'améliorer l'algorithme local de Lesk dans notre système de DL.

Notre extension considère les sens proches d'un autre en fonction d'un seuil de similarité δ_1 sur la distance cosinus avec le lemme du sens cible, et un filtrage supplémentaire sur le vecteur de sens cible, avec un seuil δ_2.

Ces deux seuils ont été paramétrés sur un corpus de développement. Ici, nous avons utilisé la tâche 7 de SemEval 2007 (Navigli *et al.*, 2007) pour fixer le meilleur ensemble de paramètres puis testé sur la tâche 13 de SemEval 2015. Nous avons échangé les corpus dans un second temps. Ce choix a été motivé pour présenter des résultats sur deux corpus de même nature, la tâche de SemEval 2007 étant très couramment utilisé dans les articles traitant de DL.

Les cinq modèles présentés dans la section 2 sont évalués séparément. Les paramètres δ_1 et δ_2 ont été estimés en les faisant varier dans les intervalles $[0.5, 0.9]$ avec un pas de 0.1.

Les résultats des meilleurs jeux de paramètres sur SemEval 2007 et SemEval 2015 sont présentés dans le tableau Table 1. La comparaison de notre méthode par rapport aux deux systèmes étalons est présentée dans le tableau Table 2.

Modèle	baroni_c		baroni_p		deps		glove		word2vec	
Paramètres	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2
Meilleurs sur SemEval 2007	0.6	0.6	0.5	0.5	0.6	0.8	0.5	0.6	0.5	0.6
Meilleurs sur SemEval 2015	0.5	0.8	0.5	0.6	0.6	0.8	0.5	0.7	0.5	0.6

TABLE 1 – Estimation des paramètres δ_1 et δ_2 sur SemEval 2007 et SemEval 2015.

Système	SemEval 2007 F1 score	SemEval 2015 F1 score
Chen *et al.* (2014)	*75.80%* [6]	
Lesk	68.70%	50.65%
Lesk étendu	78.01%	61.42%
VecLesk (baroni_c)	**75.29**%	58.02%
VecLesk (baroni_p)	73.52%	53.46%
VecLesk (deps)	73.02%	56.40%
VecLesk (glove)	73.00%	**59.01**%
VecLesk (word2vec)	73.30%	57.00%

TABLE 2 – Comparaison de nos résultats sur SemEval 2007 et SemEval 2015 pour chacun des modèles de vecteurs de mots par rapport aux étalons Lesk et Lesk étendu. Les paramètres δ_1 et δ_2 utilisés sont ceux estimés dans le précédent tableau sur **l'autre tâche**, pas celle qui est testée.

Les résultats montrent que notre extension améliore très significativement les résultats obtenus par la mesure de Lesk. L'amélioration, sur SemEval 2007, est d'environ +5% pour la pire combinaison et de +7% pour la meilleure. Sur SemEval 2015 les améliorations oscillent entre +3% et +9%. Cependant, notre extension n'atteint pas les résultats du Lesk étendu ; ceci montre que notre réseau lexical est un peu moins performant qu'un réseau créé manuellement (tel que *WordNet*).

Un point intéressant à noter est la différence de score obtenue entre les différents modèles de vecteurs de mots. Les meilleurs résultats sur SemEval 2007 utilisent la méthode de Baroni *et al.* (2014) basée sur le comptage. Ceci tend à montrer que les méthodes de création de vecteurs de mots plus "ancienne" sont parfois meilleures que les modèles prédictifs. Cependant, sur SemEval 2015, c'est la méthode GloVe qui obtient les meilleurs résultats. Ces fluctuations dans les résultats sont peut être dues aux différentes natures des méthodes de création des vecteurs de mots, ou bien aussi dues aux différents corpus sur lesquels ils ont été entrainés. En tout cas, nous montrons que quelque soit le modèle sous-jacent utilisé, notre méthode améliore systématiquement le score de Lesk.

Par ailleurs si l'on compare notre système à la meilleure méthode état de l'art à notre connaissance exploitant des ressources similaires aux nôtres (un dictionnaire et des corpus non annotés), nous observons que nos résultats sont les meilleurs sur les systèmes de DL à base de connaissances. Par ailleurs nous souhaitons souligner que les résultats présentés par Chen *et al.* (2014) sont biaisés par le fait que les paramètres optimisant leur système sont estimés sur le corpus de test. À noter que nous obtenons un score de 77.08% sur SemEval 2007 avec le meilleur jeu de paramètres appris sur cette même tâche.

6. Un biais est présent dans l'article référencé étant donné que les auteurs estiment leur paramètre $\delta \in [-0.1, 0.3]$ comparable au nôtre, mais directement durant la phase de test. Leurs scores varient entre 72.10% et 75.80% en fonction de δ.

5 Conclusion

Dans cet article nous avons présenté un modèle de vecteurs de sens, représentant tous les sens issus d'un dictionnaire. Ce modèle est construit en se basant sur les définitions du dictionnaire et sur un modèle de vecteurs de mots existant. Notre méthode consiste à sommer les vecteurs de mots présents dans la définition, pondérés en fonction de leur partie du discours ainsi que de leur fréquence.

Nous avons créé cinq modèles de vecteurs de sens, chacun reposant sur un modèle de vecteurs de mots différents. Nos modèles sont distribués librement sur la page de notre article.

Ces modèles sont ensuite utilisés comme un réseau lexical pour améliorer un système de DL basé sur l'algorithme de Lesk. Nous utilisons une méthode similaire à Banerjee & Pedersen (2002) avec comme grande différence l'utilisation d'un réseau lexical construit totalement automatiquement.

Les résultats que nous obtenons sur deux tâches de DL montrent une amélioration systématique du score par rapport à la mesure de Lesk classique (d'environ +3% à +9% selon le modèle).

Nous avons appliqué notre méthode uniquement sur la base lexicale WordNet pour évaluer nos modèles sur des tâches très connues en DL sur la langue anglaise. Cependant, notre méthode peut être appliquée très facilement à d'autres langues où les ressources sont moindres : un dictionnaire classique ainsi que des corpus non annotés sont les ressources suffisantes pour ainsi créer un système de désambiguïsation lexicale performant.

Références

BANERJEE S. & PEDERSEN T. (2002). An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing 2002*, Mexico City.

BARONI M., DINU G. & KRUSZEWSKI (2014). Don't count, predict ! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 238–247, Baltimore, Maryland : Association for Computational Linguistics.

CHEN X., LIU Z. & SUN M. (2014). A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1025–1035, Doha, Qatar : Association for Computational Linguistics.

FERRERO J., BESACIER L., SCHWAB D. & AGNÈS F. (2017). CompiLIG at SemEval-2017 Task 1 : Cross-Language Plagiarism Detection Methods for Semantic Textual Similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, Vancouver, Canada.

LEVY O. & GOLDBERG Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2 : Short Papers*, p. 302–308.

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

MILLER G. A. (1995). Wordnet : A lexical database. *ACM*, **Vol. 38**(No. 11), p. 1–41.

NAVIGLI R. (2009). Wsd : a survey. *ACM Computing Surveys*, **41**(2), 1–69.

NAVIGLI R., LITKOWSKI K. C. & HARGRAVES O. (2007). Semeval-2007 task 07 : Coarse-grained english all-words task. In *SemEval-2007*, p. 30–35, Prague, Czech Republic.

PENNINGTON J., SOCHER R. & MANNING C. D. (2014). Glove : Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, p. 1532–1543.

VIAL L., TCHECHMEDJIEV A. & SCHWAB D. (2017). Comparison of global algorithms in word sense disambiguation.

YUAN D., RICHARDSON J., DOHERTY R., EVANS C. & ALTENDORF E. (2016). Semi-supervised word sense disambiguation with neural models. In *COLING 2016*.