Sense Embeddings in Knowledge-Based Word Sense Disambiguation

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

Abstract

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.

1 Introduction

In Natural Language Processing (NLP), Word Sense Disambiguation (WSD) aims at assigning the most probable sense of a word in a document, given a pre-defined sense inventory. State of the art methods in WSD are often supervised systems, as stated by Navigli (2009), which are trained thanks to a great quantity of sense usage examples coming from sense-annotated corpora. The trained model is then used to tag a word with the sense that appears to be the most correct given its context. Unsupervised and knowledge-based methods, on the other hand, have the advantage that they require much less resource to work, and in particular no sense-annotated corpora. Hence they offer a wider coverage more easily, as they do not need to *learn* a sense from an example in order to assign it to a word. In addition, unsupervised and knowledge-based methods are generally the only usable systems to work for disambiguating another language than English. Indeed, sense-annotated corpora are very expensive resources to produce, and almost practically inexisting for every other languages [1]. For this reason, this paper will focus on a knowledge-based method, and on a novel approach that improves its performance in an unsupervised manner, by using word embeddings.

Word embeddings are a set of methods which aim to represent words as vectors. Several recent state of the art methods such as Mikolov et al. (2013)'s Word2Vec, Pennington et al. (2014)'s GloVe and Levy and Goldberg (2014)'s dependency-based vectors have proven to be very useful in many NLP tasks, like Machine Translation, Word Similarity tasks, and even in WSD, where word embeddings are also parts of some recent methods, such as Chen et al. (2014), Iacobacci et al. (2016) or Yuan et al. (2016).

In this paper, we are going to produce sense embeddings, i.e. vectors which represent senses present in the lexical database Princeton WordNet (Miller, 1995). The methodology used to create these vectors is described in section 2. Other methods exist for computing sense embeddings, such as in Iacobacci et al.

[1]Less than ten different languages have at least one corpus sense annotated with WordNet, as listed in `http://globalwordnet.org/wordnet-annotated-corpora/`

(2015) for example, who learn a distributional representation of senses through sense-annotated corpora and Word2Vec, but it presupposes a good WSD system, which is already able to sense-annotate precisely. Our sense vectors are hence evaluated by comparing their performance when used as a semantic network for a knowledge-based WSD system described in section 3.1.

The sense embeddings will be used for expanding the gloss of every sense in WordNet, by concatenating the gloss of the most related senses, in a similar way than Banerjee and Pedersen (2002)'s Extended Lesk algorithm does, but instead of considering that two senses are related because they share a lexical or semantic relation in the WordNet network, we consider two senses to be related if their vector's cosine similarity reaches a certain threshold. In the experiments in section 4, we evaluate our method on two different English all-words disambiguation tasks: first, we learn the best value of the similarity threshold on a task, then, this value is tested on the other task. The scores shown are hence the ones we obtain for the latter task, and thereafter, we perform the opposite.

The method could be applied to any language for which it exists a dictionary and a set a unannotated corpora. However, it is evaluated on an English task for at least two reasons: First, it will be easier to compare our results to other systems, since the main WSD evaluation campaigns also use WordNet. Second, we wanted to compare the benefits of different pre-existing word embeddings models to our system, and the most popular ones are trained on English corpora.

2 Creation of Sense Embeddings

Our model for representing a dictionary's sense vector relies on an existing word embeddings model, and on the sense's definition in the dictionary. In practice, our method is very similar to the one presented by Ferrero et al. (2017), who create sentence vectors for a cross-language semantic textual similarity task. Our sense vector is computed as the normalized sum of all the terms' vectors present in the sense's definition, weighed in function of their part of speech (noun, verb, adjective or adverb), and also weighed by their inverse document frequency (IDF), i.e. the inverse of the number of occurrence in the entire dictionary. More formally, we denote:

- $D(S) = \{w_0, w_1, w_2, \ldots, w_n\}$ the definition of sense S in the dictionary
- $pos(w_n) = \{n, v, a, r\}$ the part of speech of the term w_n (noun, verb, adjective or adverb)
- $weight(pos)$ the weight associated with a specific part of speech
- $idf(w_n)$ the IDF value of w_n, computed as $\log(\frac{N_{tot}}{N(w_n)})$, with N_{tot} being the total number of definitions in the dictionary (206,941 in WordNet 3.0), and $N(w_n)$ the number of definitions containing at least one occurrence of the word w_n.

The definition of the vector of the sense S, denoted $\phi(S)$ is hence the following:

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

$\phi(S)$ is then normalized, in order to be the same length as the vectors contained in the word embeddings model (generally the length is 1). The chosen POS weights are the same that Ferrero et al. (2017) used for representing English and Spanish sentences, but they can be trained as parameters of the model.

We generated five sense embeddings models, all created for every sense of WordNet 3.0, but exploiting a different word embeddings model. The five word embeddings model that we used are:

1. The pre-trained word vectors available on the Web page of the original Mikolov et al. (2013)'s Word2Vec [2]. This model was trained on about 100 billion words from Google News datasets. The vocabulary size is about 3 million words and phrases, and the vectors have a dimension of 300.
2. The pre-trained Pennington et al. (2014)'s GloVe [3], trained on 42 billion words from Common

[2] `https://code.google.com/archive/p/word2vec/`
[3] `https://nlp.stanford.edu/projects/glove/`

Crawl. The vocabulary size is about 2 million words, and the vectors have a dimension of 300.

3. The pre-trained Levy and Goldberg (2014)'s dependency-based word embeddings [4]. The training was done on Wikipedia. The vocabulary size is about 175,000 words, and the dimension is 300.

4. The best *predict* vectors created in Baroni et al. (2014) [5]. The vectors have a dimension of 400 and the vocabulary size is about 300,000.

5. Finally, the best *reduced count* vectors also created in Baroni et al. (2014) [5], of dimension 500, and with the same vocabulary than the previous model.

In the experiments section, we compare the performance of each of the embeddings model in our WSD system extension. All five sense embeddings models are publicly released on our GitHub[6].

3 Evaluation in Knowledge-Based WSD

The generated vector representation of senses will be evaluated on a WSD task, as a supplementary resource for a knowledge-based method. The idea is to use the embeddings model as a semantic network, which is able to fetch senses related to another sense. These related senses will then contribute to the similarity measure used in the WSD system, in the same manner than Banerjee and Pedersen (2002)'s usage of the semantic network integrated in WordNet.

3.1 Disambiguation Algorithm

The knowledge-based WSD system that is used to compare the performance of the semantic network is built as two pieces: the Local Algorithm, which computes a score of similarity for a pair of senses, and the Global Algorithm, which searches for the best combination of senses at the document level, using the local algorithm.

The global algorithm is a heuristic that will avoid to compute every possible combination of senses in the document, because this would lead to an uncomputable number of calls to the local algorithm (the average number of senses per word, to the power of the number of words in the document).

The heuristic used in our system is an implementation of the Cuckoo Search Algorithm, as research done by Vial et al. (2017) on Global Algorithms shows that it is among the best global algorithm for this kind of knowledge-based WSD system. Our implementation uses as parameters a single cuckoo, a Levy location of 5 and a Levy scale of 0.5, as described as the best parameters in Vial et al. (2017). In all of our experiments, we set the number of iterations to be a very large number (300,000), so the results do not differ notably from an execution to another. In addition, we compute the mean of 10 complete executions when giving a score, and we ensure a low standard deviation (generally < 0.1) so the end result is stable and reliable.

The local algorithm is the central element of the system and this is where the usage of the semantic network will occur. As a baseline, we are going to use the original Lesk measure, also called gloss overlap measure. This algorithm returns, as a similarity score, the number of words in common in the two senses' definition. Formally, if we denote $D(S) = \{w_1, w_2, \ldots, w_n\}$ the definition of S, then the Lesk measure between sense S_1 and sense S_2, denoted $Lesk(S_1, S_2)$ is the following:

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

As a second baseline, and in order to compare the quality of our produced semantic network, we will also use the Extended Lesk measure, described in Banerjee and Pedersen (2002). This measure consider not only the definition of the target sense, but also the definition of every sense that have a relation to the

[4]https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/
[5]http://clic.cimec.unitn.it/composes/semantic-vectors.html
[6]https://github.com/getalp/WSD-IWCS2017-Vialetal

target sense (hyperonymy, hyponymy, etc.). Let's denote $rel(S)$ the set of senses related to S, through an explicit link in WordNet, then the Extended Lesk measure between sense S_1 and sense S_2, denoted $ExtLesk(S_1, S_2)$ is the following:

$$ExtLesk(S_1, S_2) = |(D(S_1) \cup D(rel(S_1))) \cap (D(S_2) \cup D(rel(S_2)))|$$

3.2 Gloss Expansion Through Sense Embeddings

Our method for using our sense embeddings model produced in section 2 is to extend the Lesk algorithm in a way similar to Banerjee and Pedersen (2002), that is by also considering the related senses when computing the similarity of two senses.

Because our sense vectors are created from the vectors of the terms used in the dictionary's glosses, the impact of these terms is huge. This is why for our expansion, we will take into account the most similar senses not using solely the cosine similarity, because this would blindly fetch some unrelated senses that are described using the same kind of wording. We also filter on the lemma's vector of the target sense in the word embeddings space, so the fetched senses will also be close to the *idea* of the sense's lemma, as captured by the word embeddings model. As a recall, operations between a sense vector and a word vector are possible because the sense vector is, in essence, only a weighted sum of word vectors.

Let's denote $rel(S, \delta_1, \delta_2)$ the set of senses related to S. δ_1 is the threshold cosine similarity value on the **lemma**'s vector, below which we do not consider a sense as related. δ_2 is the threshold cosine similarity value on the **sense**'s vector, below which we do not consider a sense as related. $Senses$ is the set of all the senses in the dictionary. $cosine$ is the cosine similarity operator. The formula for computing $rel(S, \delta_1, \delta_2)$ is the following:

$$rel(S, \delta_1, \delta_2) = \{S' \mid cosine(\phi(lemma(S)), \phi(S')) > \delta_1, cosine(\phi(S), \phi(S')) > \delta_2\}$$

Finally, the formula for our new local algorithm for the WSD system, which we will be denoted as $VecLesk(S_1, S_2, \delta_1, \delta_2)$, can be written almost exactly as the Extended Lesk:

$$VecLesk(S_1, S_2, \delta_1, \delta_2) = |(D(S_1) \cup D(rel(S_1, \delta_1, \delta_2))) \cap (D(S_2) \cup D(rel(S_2, \delta_1, \delta_2))|$$

In the following section, we are going to evaluate this new local algorithm on two WSD all-words tasks, in regards with the Lesk and the Extended Lesk baselines.

4 Experiments

In order to see how our sense embeddings model performs as a semantic network, we use it for a lexical expansion of the dictionary's glosses, for improving the Lesk local algorithm of our WSD system.

Our expansion considers the related senses regarding two cosine similarity threshold: δ_1, filtering out senses based on their similarity with the target sense's lemma vector, and δ_2, filtering out senses based on their similarity with the target sense vector.

These two parameters have to be set in some way, so we chose two WSD tasks: SemEval 2007 task 7 (Navigli et al., 2007), and SemEval 2015 task 13 (Moro and Navigli, 2015), and we estimated the best set of parameters on a task, then tested this set of parameters on the other. The reason why we chose these two tasks is that they are of the same nature, i.e. both all-words WSD tasks, and that the task 7 of SemEval 2007 is largely used in most WSD articles, so it is easier to put the results in perspective.

All five word embeddings models mentioned in section 2 are evaluated separately. And the parameters δ_1 and δ_2 have been estimated by testing every values in the range $[0.5, 0.9]$ with steps of 0.1.

The results of the best parameters estimation on SemEval 2007 task 7 and on SemEval 2015 task 13 is in Table 1. The comparison of our method on both tasks in regards with the baselines and a state of the art system is in Table 2.

Word Embeddings Model	baroni_c		baroni_p		deps		glove		word2vec	
Parameters	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2	δ_1	δ_2
Best values on SemEval 2007	0.6	0.6	0.5	0.5	0.6	0.8	0.5	0.6	0.5	0.6
Best values on 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 of parameters δ_1 and δ_2 on SemEval 2007 task 7 and SemEval 2015 task 13.

System	SemEval 2007 F1 score	SemEval 2015 F1 score
Chen et al. (2014)	*75.80%*[7]	
Lesk baseline	68.70%	50.65%
ExtLesk baseline	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: Comparison of our results on SemEval 2007 task 7 and SemEval 2015 task 13 for each word embeddings model used, in regards with the Lesk and Extended Lesk baselines and a state of the art method that uses similar resources than us. The parameters δ_1 and δ_2 used in VecLesk are taken from the parameter estimation of Table 1 **from the other task**, not the one that is tested.

The results show that our extension improves greatly the score of the Lesk measure, which is at least around +5% for the worst combination of word embeddings model and parameters, on SemEval 2007, and around +7% for the best combination. On SemEval 2015, the worst extension still improves the score by +3%, and the best one gives +9%. Our extension does not reach the score of the Extended Lesk baseline however. Which is a sign that our semantic network is probably less relevant than the explicit links found in WordNet.

An interesting data is the difference of score obtained by the different word embeddings model. The best result on SemEval 2007 uses Baroni et al. (2014)'s count vectors, and the score is 2% higher than the second best word embeddings model's score (using Baroni et al. (2014)'s predict vectors). This tends to show that the "older" approach of word embeddings creation, i.e. counting-based vectors, are in some cases a better choice than the predicting models. However, on SemEval 2015, the best model is GloVe. Baroni's count vectors is a close second though. The fluctuation of the results in function of the model used may be because of the different natures of the word embeddings models, or due to the different corpora they were trained on. In any cases, our extension works with any model, systematically raising the score from the Lesk baseline.

The comparison of our system to the best existing method to our knowledge that uses the same kind of resources than us (i.e. a dictionary and unannotated corpora), shows that our extension achieves state of the art results on methods using such few resources. The score achieved by Chen et al. (2014) has to be treated with caution, because they learned a threshold parameter δ similarly to us, for their construction of vectors, however they estimated their best parameter and tested on the same corpus, leading to an obvious bias. Note that in the same conditions, when we use the best set of parameters learned on this same task, our method reaches a score of 77.08% on SemEval 2007.

[7]The referenced article's score is biased, since the authors' system learns a parameter $\delta \in [-0.1, 0.3]$ comparable to our, but directly during the testing phase. The score ranges from 72.10% to 75.80% depending on the value of their δ.

5 Conclusion

In this article, we created a Sense Embeddings model, representing every sense of a dictionary, based on the words contained in their gloss and using an existing Word Embeddings model. Our method of construction essentially computes the sum of the gloss terms' vectors, weighted in function of their part of speech and their inverse frequency. We created five sense embeddings models, each one of them relying on different word embeddings model. They are available publicly on our GitHub[8].

The models are then used as a semantic network for improving a knowledge-based WSD system, based on the Lesk algorithm. The idea is to take into account the closest senses to a target sense in our semantic network, in order to disambiguate it, in the same manner as Banerjee and Pedersen (2002) do using the related senses information built in WordNet.

The resulting extended WSD system performs systematically better than its unextended baseline counterpart, improving the score from about +3% for the worst extension, to about +9% for the best one.

This article uses WordNet as a dictionary, and evaluations are performed on two English all-words WSD tasks, because it is easier to compare the performance and the robustness of the method, as the majority of the researches in WSD uses this language. However, the whole process of sense embeddings creation and Lesk extension can be easily adapted to many language, requiring only a set of unannotated corpora, and a typical dictionary, thus, giving the possibility to create an efficient WSD system, even for a poorly resourced language.

[8]`https://github.com/getalp/WSD-IWCS2017-Vialetal`

References

Banerjee, S. and T. Pedersen (2002, February). An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing 2002*, Mexico City.

Baroni, M., G. Dinu, and Kruszewski (2014, June). 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)*, Baltimore, Maryland, pp. 238–247. Association for Computational Linguistics.

Chen, X., Z. Liu, and M. Sun (2014, October). A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1025–1035. Association for Computational Linguistics.

Ferrero, J., L. Besacier, D. Schwab, and F. Agnès (2017, August). 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.

Iacobacci, I., M. T. Pilehvar, and R. Navigli (2015). Sensembed: Learning sense embeddings for word and relational similarity. In *In Proceedings of ACL*, pp. 95–105.

Iacobacci, I., M. T. Pilehvar, and R. Navigli (2016, August). Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, pp. 897–907. Association for Computational Linguistics.

Levy, O. and Y. Goldberg (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*, pp. 302–308.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc.

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

Moro, A. and R. Navigli (2015, June). Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado, pp. 288–297. Association for Computational Linguistics.

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

Navigli, R., K. C. Litkowski, and O. Hargraves (2007, June). Semeval-2007 task 07: Coarse-grained english all-words task. In *SemEval-2007*, Prague, Czech Republic, pp. 30–35.

Pennington, J., R. Socher, and C. D. Manning (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Vial, L., A. Tchechmedjiev, and D. Schwab (2017). Comparison of global algorithms in word sense disambiguation. *CoRR abs/1704.02293*, 1–22.

Yuan, D., J. Richardson, R. Doherty, C. Evans, and E. Altendorf (2016). Semi-supervised word sense disambiguation with neural models. In *COLING 2016*.