Efficient Unsupervised Word Sense Induction, Disambiguation and Embedding

Behrouz Haji Soleimani, Habibeh Naderi, Stan Matwin
Institute for Big Data Analytics
Faculty of Computer Science
Dalhousie University, Halifax NS, Canada
{behrouz.hajisoleimani, habibeh.naderi, st837183}@dal.ca

Abstract

We present an efficient word sense disambiguation and embedding algorithm that learns multi-prototype sense vectors to accommodate different meanings of words. Our method provides both sparse sense vectors to be used for word sense induction, and dense sense embeddings to be used in training of machine learning models for downstream NLP tasks. We disambiguate the ambiguous words by clustering their context words. However, unlike previous methods where each context is assigned to one sense, we use a soft clustering approach to learn the distribution of context words over different senses and use it to break the global context vectors into multiple sense-specific sparse vectors. After constructing the sense-word relation matrix from the word-word co-occurrence matrix, we use it for learning sense embeddings using a fast matrix factorization approach. Our experiments on SemEval 2013 competition data show that our method achieves the state-of-the-art accuracy in a more efficient way.

1 Introduction

Continuous-valued word representations have become very popular since they improve the performance of many natural language processing applications. These distributional representations are learned from large text corpora and can explain many semantic properties and relationships between concepts represented by words. Many neural-network based approaches [4, 10, 15, 16] as well as count-based [19, 1] and matrix factorization-based methods [13, 21] have been proposed for learning distributed word representations. The Skip-Gram model was very successful so that many improvements and variants are proposed for it. For instance, FastText [6] enriches the Skip-Gram word embeddings with sub-word information by considering character n-grams of different lengths and representing words as the sum of their n-gram vectors.

Conventional word embeddings learn a unique representation for each word and therefore, they cannot deal with word ambiguity which is an important property of the natural language. For example, the word "right" may refer to a direction or to being correct depending on the context. Consequently, the learned vector representation is either dominated by the most frequent meaning or it is a weighted average of all the meanings [2]. Clearly neither case is desirable for practical applications. Since word representations have been used as word features in many NLP tasks including dependency parsing [8], named-entity recognition [22], emotion recognition from tweets [17] and sentiment analysis [14], using a multi-prototype representation can potentially increase the performance of such representation-based approaches.

Several approaches have been proposed for learning sense-specific word vectors. [20] proposed a clustering based approach in which context words are clustered in order to produce groups of similar context vectors. An average "prototype" vector is then computed separately for each cluster,

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

producing a set of sparse sense vectors for each word. [11] uses a similar clustering approach to cluster sparse TF-IDF context vectors to create a fixed number of senses for each word. Then they relabel each word token with the clustered sense before learning embeddings. SenseGram [18] and AdaGram [3] are among the most successful word sense embedding methods. AdaGram is a Bayesian extension of the Skip-Gram model and provides Word Sense Disambiguation (WSD) functionality based on the induced sense inventory. SenseGram transforms word embeddings to sense embeddings via graph clustering and uses them for WSD.

In this work, we present a fast unsupervised multi-prototype sense representation method. Instead of using traditional clustering approaches which assign each context word to a specific sense, we use an efficient soft clustering approach where contexts are assigned to different senses according to their membership values. For any ambiguous word w, most of its context words are common across all of its senses. Therefore, learning the distribution of words across senses is an intuitive and natural choice. Moreover, our algorithm outputs both sparse and dense representations for senses. The sparse representations are interpretable and can be used for word sense induction. The correct sense of the word can be induced by comparing the usage of the word in a given sentence with the sparse sense vectors and picking the most similar sense. The dense representations can be used as word features in training machine learning models for any desired NLP application such as sentiment analysis, translation, etc.

2 Proposed Method

Similar to word embeddings, sense embeddings should also be learned from a large text corpus such as Wikipedia or Common Crawl data. In our method, we first calculate the global symmetric word-word co-occurrence counts matrix X by moving a t-sized context window over the corpus. Assuming that the corpus contains n unique words (i.e. vocabulary $V = \{w_1, w_2, \ldots, w_n\}, |V| = n$), then the co-occurrence counts X is a highly sparse $n \times n$ matrix. This co-occurrence matrix contains global statistics about word relations and is the primary source of information for many word embedding algorithms such as GloVe [19]. Each row i of this matrix is the global context vector $\overrightarrow{x}_i$ of a particular word w_i which represents all possible context words that are co-occurred with w_i in the corpus. Please note that each global context vector $\overrightarrow{x}_i$ is the algebraic sum of all context vectors of all the meanings of w_i. In section 2.2 we will break these vectors down into multiple context vectors, one for each sense.

2.1 Refining the co-occurrences and extracting word relations

Many of the co-occurrences in X are meaningless. For instance, the words "also" and "their" co-occur thousands of times in Wikipedia but this should not be considered as a high degree of association. Pointwise Mutual Information (PMI) is an information theoretic measure that can be used for finding collocations or associations between words [9] and is widely used in count-based and matrix factorization based word embeddings. For any word pair (w_i, w_j), PMI is defined as the log ratio between their joint probability and product of their marginal probabilities:

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \tag{1}$$

If two words co-occur more often than being independent then their PMI will be positive, and if they co-occur less frequent than being independent then their PMI will be negative. Hence, a commonly accepted approach is to use Positive PMI (PPMI) matrix by replacing all the negative values with 0, $PPMI(w_i, w_j) = \max(PMI(w_i, w_j), 0)$. Therefore, unlike many sense disambiguation methods that use TF-IDF approach which is more suitable for document-term matrices, we take the PPMI of the co-occurrence matrix X that results in a more refined and accurate word relation matrix P.

2.2 Word Sense Disambiguation: Obtaining Sparse Sense Vectors

For any ambiguous word w_i we break its refined context vector $\overrightarrow{p}_i$ (i.e. i-th row of the PPMI matrix P) into multiple context vectors, one for each sense of the word. Assuming that word w_i has k_i different senses $\{s_{i1}, s_{i2}, \ldots, s_{ik_i}\}$, we learn a sparse context vector $\overrightarrow{s}_{ij}$ for each sense s_{ij} in such

a way that $\sum_{j=1}^{k_i} \overrightarrow{s}_{ij} = \overrightarrow{p}_i$. This will ensure that sum of all context vectors from different senses equals to the global context vector of the word.

We break the context vector $\overrightarrow{p}_i$ of the word w_i by clustering the context vectors of its context words. Let $L_i = \{l_1, l_2, \ldots, l_m\}$ be the set of all non-zero entries in $\overrightarrow{p}_i$ which is the set of indices of all context words of w_i. Then, we select a slice C_i of the matrix P where only rows corresponding to L_i are selected. C_i will be a $m \times n$ matrix representing the context vectors of the context words of w_i. Our aim is to learn the distribution of context words over senses and we achieve this by applying a soft clustering. We need to do this in an efficient way since the clustering has to be done for each ambiguous word separately.

Fuzzy C-means clustering is not applicable in this case since the input matrix is high dimensional and therefore, all the centroids in fuzzy clustering will converge to the center of gravity of the entire data distribution [23]. We propose to use Non-negative Matrix Factorization (NMF) in order to cluster the columns of C_i. NMF decomposes the matrix $C_i = WH$ in the form of multiplication of two matrices $W_{m \times k}$ and $H_{k \times n}$ by minimizing the reconstruction error of the original matrix constraining all elements of all three matrices to be non-negative. NMF can be used to cluster the columns of the input matrix and $H_{k \times n}$ is known to produce membership values of n points to k clusters [12]. We use WordNet lexical database to obtain the number of senses k_i for each ambiguous word w_i and apply NMF with k_i components. Finally, we normalize membership values for each context word to sum 1 and multiply the membership matrix by the global context vector $\overrightarrow{p}_i$ to obtain the sparse sense vectors $\overrightarrow{s}_{ij} = \overrightarrow{h}_j \odot \overrightarrow{p}_i$. Normalizing the columns of H ensures $\sum_{j=1}^{k_i} \overrightarrow{s}_{ij} = \overrightarrow{p}_i$.

Latent Dirichlet Allocation (LDA) and topic modeling [5] also seems to be a good choice for the soft clustering of contexts as it can learn the distribution of context words in different topics (or senses in this case). However, LDA is resource intensive and it was computationally intractable to disambiguate all words in our experiments using LDA. For instance, disambiguating a single word with around 50,000 context words (i.e. LDA on a matrix of 50,000 rows) takes more than 1.5 hours while NMF on the same matrix ends in 2 minutes.

Further improving the efficiency Although NMF using coordinate descent optimization is pretty fast, we take it a step further and make our algorithm faster by only taking the initialized value of H. We use the Non-Negative Double Singular Value Decomposition (NNDSVD) method [7] which is often used to initialize W and H matrices for NMF and is shown to provide a very good starting point [7]. Therefore, we just use the output of NNDSVD without optimizing it further to reduce the runtime to the minimum. Our experiments did not show a significant difference between the accuracy of the two methods, thus, we pick the fastest and report the results of NNDSVD in section 3.

2.3 Word Sense Induction Using the Sparse Sense Representation

The sparse sense vectors can be used to infer the correct sense of a word in a given sentence based on its context words. Then we can select the corresponding sense embedding and use it for the underlying NLP task. For the purpose of word sense induction, we build the context vector of the query sentence $\overrightarrow{q}_i$ and use Cosine similarity to find the most similar sense vector $\{\overrightarrow{s}_{i1}, \ldots, \overrightarrow{s}_{ik_i}\}$:

$$j^* = \arg\max_j \frac{\overrightarrow{s}_{ij} . \overrightarrow{q}_i}{|\overrightarrow{s}_{ij}| \times |\overrightarrow{q}_i|} \tag{2}$$

2.4 Word Sense Embedding: Obtaining Dense Representations

We augment all the sparse sense vectors of all ambiguous words to the P matrix and build a global sense-word co-occurrence matrix S and use it for the embedding. After constructing the sense-word relation matrix, one can use GloVe [19] optimization or any type of matrix factorization approaches [13, 21] in order to obtain the sense embeddings. For the purpose of efficiency we use a sparse Singular Value Decomposition (SVD) on the sense-word relation matrix $S = U\Sigma V^T$ and take the first d singular vectors of U as our dense representations. This sparse factorization uses the efficient Lanczos iterative sparse eigensolver to obtain the top d singular vectors corresponding to the largest singular values and is shown to capture semantic similarities pretty well when applied on co-occurrence type relational matrices [21].

Table 1: Nearest neighbors of sample words in our sense embedding vector space

Word	Nearest neighbors
program_1	mentoring, internships, scholarships, educational, volunteering, trainings
program_2	software, implementations, computing, data, protocol, algorithms, automated
program_3	aired, hosted, televised, telecast, christmas, filmed, wrestling, announcer
square_1	plaza, street, apartments, palace, town, cafe, landmark, mall
square_2	rectangular, arched, octagonal, cornice, triangular, hexagonal, gable
square_3	km, meters, households, kilometers, hectares, ft, sq, islander, wingspan
interest_1	interested, attention, fascination, curiosity, attraction, hobby, keen, enthusiasm
interest_2	ventures, assets, acquisitions, investments, contracts, subsidiaries, leases, holdings
interest_3	romantically, girlfriend, friend, relationship, friendship, actress, partner

Table 2: Comparison of our method's performance to participants of the SemEval 2013 Task 13 and two systems based on word sense embeddings (AdaGram and SenseGram)

Model	Jaccard	Tau	WNDCG	Fuz. NMI	Fuz. B-Cub.	Average
AI-KU	0.197	0.620	**0.387**	0.065	0.390	0.3318
AI-KU (rem5-add1000)	0.245	0.642	0.332	0.039	0.451	0.3418
Unimelb (50k)	0.213	0.620	0.371	0.060	0.483	0.3494
UoS (top-3)	0.232	0.625	0.374	0.045	0.448	0.3448
SenseGram, p=2, d=100	0.197	0.615	0.291	0.011	**0.615**	0.3458
AdaGram, α=0.05, d=100	**0.274**	**0.644**	0.318	0.058	0.470	0.3528
Our Method	0.215	0.610	0.335	**0.072**	0.560	**0.3584**

3 Evaluation

We used Wikipedia dump of April 2018 as our training corpus which has about 2.2 billion tokens. After a few preprocessing steps we applied our algorithm to learn both sparse and dense representations for all the senses of all words.

Qualitative Evaluation Table 1 shows the nearest neighbors of the different senses of words "program", "square", and "interest" in our sense embedding vector space. As we can see it can disambiguate different senses very well. For instance, the three senses of word "program" correspond to training/educational programs, software programs, and television programs which are quite distinct.

Quantitative Evaluation We evaluated our learned sense vectors on the SemEval 2013 Task 13 word sense disambiguation competition data. This dataset contains 50 ambiguous words and about 100 example sentences per each word for a total of 4664 instances. The algorithms have to infer the correct sense for each of the instances. For many test sentences, multiple senses are applicable with different levels of appropriateness and this enables various scoring systems. We have used all 5 official scoring functions released by the competition organizers (Jaccard Index, Positional Tau, Weighted NDCG, Fuzzy NMI, and Fuzzy B-Cubed) and compared our algorithm with the top winning teams as well as the current state-of-the-art approaches AdaGram and SenseGram in table 2. Looking at the performances of different algorithms in the table we can see that our method has achieved the best overall accuracy, though it is very close to the current state-of-the-art.

4 Conclusion

In this work, we analyzed word sense disambiguation and embedding methods in NLP and proposed an efficient alternative that achieves state-of-the-art accuracy. Our method uses a soft clustering approach to group context words which has a clear advantage over conventional context clustering methods by learning a more accurate distribution of words across different meanings. In fact, it splits the global co-occurrence distribution into multiple distributions over the senses of words. We provide both sparse and dense representations for each sense that can accommodate every NLP need from word sense induction (i.e. inferring the correct sense of a word given a query sentence) to training machine learning models using dense vector space representations. Qualitative and quantitative evaluations show the effectiveness of the proposed method.

References

[1] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[2] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association of Computational Linguistics*, 6:483–495, 2018.

[3] Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial Intelligence and Statistics*, pages 130–138, 2016.

[4] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. *Neural Probabilistic Language Models*, pages 137–186. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[7] Christos Boutsidis and Efstratios Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[8] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[9] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[10] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[11] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[12] Da Kuang. *Nonnegative matrix factorization for clustering*. PhD thesis, Georgia Institute of Technology, 2014.

[13] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[14] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[17] Habibeh Naderi, Behrouz Haji Soleimani, Saif Mohammad, Svetlana Kiritchenko, and Stan Matwin. Deepminer at semeval-2018 task 1: Emotion intensity recognition using deep representation learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 305–312. Association for Computational Linguistics (ACL), 2018.

[18] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 174–183. Association for Computational Linguistics, 2017.

[19] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[20] Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics, 2010.

[21] Behrouz Haji Soleimani and Stan Matwin. Spectral word embedding with negative sampling. In *AAAI Conference on Aritificial Intelligence, AAAI*, pages 5481–5487, 2018.

[22] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[23] Roland Winkler, Frank Klawonn, and Rudolf Kruse. Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications (IJFSA)*, 1(1):1–16, 2011.