Generating vector representation for words using semantic graphs

Abstract

Natural language processing tasks are becoming more important. While word2vec pioneered a way of representing words, it's trained using conventional word corpus. Semantic graphs holds more information on relation between words than corpus. The word relations present in those graphs can be used to reach better word representation. This Thesis aim to provide a better way to represent words as vectors. Using semantic graphs and the relation between words to embed words to vector space with a better representation word quality.

1 Introduction

With the rise of handheld devices and robotic machinery NLP tasks are becoming more relevant, and the need to develop better models require better representation of the input language sources. For years the supervised techniques for NLP has dominated the field of research. The rapid development of linguistics in the second half of the twentieth century along with the introduction of machine learning made probabilistic models prominent in all NLP research. The problem with those tasks were the need to have a manually crafted feature for the input data, along with ground truth.

Recently the research scene has partially shifted into the deep learning and input representation became crucial for the creating such models. In 2013 [MSC+13] introduced word2vec, neural network models to generate vector space embedding for words in input corpus. The methods led to increasing research in the areas of word embedding. Using lexical information from ontology graphs could lead to better word embedding. The lexical information would result in words having similar meaning and/or lexical relations to have relatively close vector representations than words that are totally different, also it would result in words that are not similar but reside in the same graph neighborhood to have close representation.

2 Related Work

Several research groups have been working on the problem of converting a graph into vectors. The research ranges from pure mathematical research for representing graphs as vectors to NLP tasks that needs to embed words from a lexical graph into vector space. In this section we are considering a subset of this work that is related to the task in hand.

In their paper [NCV+17] purposed a method for creating a vector representation of a graph by aggregating representations from sub-graphs. Building on their previous paper [NCC+16]. The algorithm begins by splitting the graph into several sub graphs similar to [SSvL+11], then it proceed to treat the sub graphs the same way as documents in [LM14]. A graph representation is then created from the several sub-graphs. This representation is for the whole graph not for individual nodes/words. This representation can then be used to tackle graph classification tasks.

Combining random walks with feature learning [PAS14], [GL16], and [GSA15] all created algorithms that start by creating a context for nodes/words using random walks combines with stochastic search strategy. This context is then used in different variation of word2vec to generate vector representation. All of the 3 methods uses hierarchical softmax along with negative sampling as in the original word2vec work.

One more method that is not using any analogy to word2vec is BLM in [IB15]. It works by calculating probability to each edge between two nodes and then using maximum likelihood and Noise contrastive estimation [Dye14] to representation calculation. Another method from classical linear algebra is Spectral Graph Analysis or Graph PCA [AM12] in which vector representation for the graph is generated using graph Laplacian matrix and Commute Time Distance transformation.

3 Baseline Experiment Description

To create a baseline for future work assessment, an initial experiment to be conducted. The initial experiment consists of replicating each of the following method:

- Spectral Graph Analysis (Grpah PCA) [AM12]
- Bilinear Link Model (BLM) [IB15]

- Random walks [GSA15] [1]

Creating embedding from each of the following lexical graphs:

- WordNet lexical database of English [2]
- Serelex lexico-semantic network [3]

And test their results on two standard test datasets :

- WS-353 [FGM$^+$01]
- SimLex-999 [HRK15]

For the BLM and Graph PCA the model will be trained on the two lexical graphs, then it will be used to generate vector embedding for each of the word pairs in the test datasets. The cosine distance between each pair's vector representations will be calculated and compared with the benchmark included with the original dataset by using Pearson correlation coefficient.

For the Random walks, context will be generated for each word on the training lexical graphs using the random walkers. This context will be then feed to word2vecf along with the word vocabulary. The final word embedding will be tested the same way as used with BLM and Graph PCA.

3.1 Evaluation

Both of the evaluation word similarity tasks used for evaluation consists of pairs of words each with a similarity rating. After training each of the models, word representation for each word in the similarity tasks will be generated. After that cosine similarity for the vector representation each pair of words will be calculated, and used as similarity rating. The similarity rating from the trained models will be compared to the similarity ratings from the similarity tasks by means of spearman's correlation coefficient.

The final output of the experiment will be a set of correlation coefficients that describe how much does those three methods compare to the original benchmarks.

[1]https://github.com/tigvarts/BLM
[2]http://wordnet.princeton.edu/
[3]http://cental.fltr.ucl.ac.be/team/ panchenko/data/serelex

		Serelex		WordNet	
Method	Task	Correlation	P-Value	Correlation	P-Value
BLM	WS-353	0.0376	0.4993	0.4143	1.88e-14
	SimLex-999	0.0026	0.9426	0.5149	6.35e-50
Random Walks	WS-353	-0.0888	0.1097	0.1509	0.00677
	SimLex-999	0.0214	0.5657	0.1125	0.0025
DeepWalk	WS-353	0.01797	0.747	-0.07180	0.1995
	SimLex-999	-0.083	0.0264	0.05297	0.1560
Direct Connection	SimLex-999	0.326	3.23e-19	0.3160	3.93e-18
	WS-353	0.587	1.38e-31	0.05997	0.2841

Table 1: Similarity tasks correlation score

3.2 Results

The first trial was using the Serelex lexico-semantic network. The network was feed to the BLM algorithm which generated vector representation for each word in the network. After that for each pair of words[4] from the two similarity tasks, words representation was extracted, then cosine similarity was calculated. The resulting list of similarity ratings for each task was then compared to the gold standard and the correlation results are shown in table 3.2.

3.3 Nearest Neighbours

To help see the data representation generated by each of the previously stated methods. A list of the 10 nearest neighbours for 5 randomly selected nodes is listed in table 3.3

4 Future Work: A New Model

This section describes a new model that will be developed in the thesis. The other models listed above will be used as the baselines.

The main difference of the model is the use of the weights, and the fact that it tries to exactly reconstruct the input graph in the objective in the vector space. In that respect the model is similar to BLM, but the latter use no weights (check!).

The objective function of the model is:

[4]421 pair of words were dropped from the two tasks, because at least one of the two words were not present in the Serelex network

Word	Method	5-NN Words
Random Walks	feline	textphone - morphine - penknife - estonia - environmental technology
deepwalks		exercise bike - deneb - lazio - heartbeat - shortness of breath
BLM		ape - opposums - dromedary - jiulongensis - rhesus monkey
Random Walks	reservation	mortality - pogues - hanson - condor - computer study
deepwalks		faith - shape - automatic - handbook - mail
BLM		rental - purchase - loan - rent - licence
Random Walks	book	xmas - concierge - seaforth - moorland - confounder
deepwalks		stoloniferum - londoncateringstaff - cummer - khean - cinepak
BLM		form - game - art - film - unit
Random Walks	money	magicicada - psychiana - damicornis - avccam - dudleys
deepwalks		fine - finance - general - electric - supermarket
BLM		idea - entertainment - festival - version - institution
Random Walks	image	bauhaus - hard times - screenwriter - km - blowback
deepwalks		rome - landscape - teaching - remain - prince
BLM		song - power - agency - set - style

$$J(\mathbf{X}) = \sum_{i \in V} \sum_{j \in V} (w_{ij} - \mathbf{x}_i \cdot \mathbf{x}_j)$$

where w_{ij} is the edge weight between vertices i and j in the set of graph vertices V, and $\mathbf{x}_i$ is a vector representing vertex i.

We will also explore a variation of this model where the original set of vertices V will be expanded into the set V' by taking into account of the vertices of the second order (the transitive paths from i to k through j). Such transitive weights w_{ik} will be computed as a product of weights to down-weight such second-order training examples:

$$w_{ik} = w_{ij} * w_{jk}.$$

We will also try other (more aggressive) ways to down-weight second-order training examples. It is very important to note that for both approaches mentioned above, the weights will be normalized to the scale $[0; 1]$: we will experiment with normalization which makes them a proper probability distribution (sum of weights of related vertices of i is 1), but also with strategy where the weight of the most similar vertex of i is 1.

The model will be optimized using the stochastic gradient descent algorithm. We can also explore negative sampling (noise contrasting estimation, NCE), to make the learning procedure faster. The model will be implemented using the TensorFlow framework.

References

[AM12] James A. Albano and David W. Messinger. Euclidean commute time distance embedding and its application to spectral anomaly detection, 2012.

[Dye14] Chris Dyer. Notes on noise contrastive estimation and negative sampling. *CoRR*, abs/1410.8251, 2014.

[FGM$^+$01] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited, 2001.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[GSA15] Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. Random walks and neural network language models on knowledge bases. In *HLT-NAACL*, 2015.

[HRK15] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with genuine similarity estimation. *Comput. Linguist.*, 41(4):665–695, December 2015.

[IB15] Oleg U. Ivanov and Sergey O. Bartunov. *Learning Representations in Directed Networks*, pages 196–207. Springer International Publishing, Cham, 2015.

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[NCC$^+$16] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *CoRR*, abs/1606.08928, 2016.

[NCV$^+$17] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017.

[PAS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014.

[SSvL$^+$11] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, November 2011.