Integrating Lexical Knowledge in Word Embeddings using Sprinkling and Retrofitting

Aakash Srinivasan[1]*, **Harshavardhan Kamarthi**[2]*, **Devi Ganesan**[2], **Sutanu Chakraborti**[2]
[1]Dept. of Computer Science, University of California, Los Angeles
[2]Dept. of Computer Science and Engineering, Indian Institute of Technology Madras
Email: {s.aakash3431, harshavardhan864.hk}@gmail.com, {gdevi, sutanuc}@cse.iitm.ac.in,

Abstract

Neural network based word embeddings, such as Word2Vec and GloVe, are purely data driven in that they capture the distributional information about words from the training corpus. Past works have attempted to improve these embeddings by incorporating semantic knowledge from lexical resources like WordNet. Some techniques like retrofitting modify word embeddings in the post-processing stage while some others use a joint learning approach by modifying the objective function of neural networks. In this paper, we discuss two novel approaches for incorporating semantic knowledge into word embeddings. In the first approach, we take advantage of Levy et al's work which showed that using SVD based methods on co-occurrence matrix provide similar performance to neural network based embeddings. We propose a sprinkling technique to add semantic relations to the co-occurrence matrix directly before factorization. In the second approach, WordNet similarity scores are used to improve the retrofitting method. We evaluate the proposed methods in both intrinsic and extrinsic tasks and observe significant improvements over the baselines in many of the datasets.

1 Introduction

Neural Network based models (Mikolov et al., 2013a; Pennington et al., 2014) have been hugely successful in generating useful vector representation for words which preserve their distributional properties in a given corpora. Improving the quality of word embeddings have led to better performance in many downstream language tasks. Considering the widespread uses of word embeddings, there have been a lot of interest in improving the quality of these embeddings by leveraging lexical knowledge such as synonymy, hypernymy, hyponymy, troponymy and paraphrase relations. This is accompanied by the availability of large scale lexical knowledge available in WordNet (Miller, 1995) and Paraphrase Database (PPDB) (Ganitkevitch et al., 2013).

In this paper, we propose two simple yet powerful approaches to incorporate lexical knowledge into the word embeddings. First, we propose a matrix factorization based approach which uses the idea of 'sprinkling' (Chakraborti et al., 2006, 2007) semantic knowledge into the word co-occurrence matrix. Second, we identify the weaknesses of the retrofitting model (Faruqui et al., 2014) and propose a few modifications that improves the performance. We demonstrate the strength of the proposed models by showing significant improvements in two commonly used intrinsic language tasks - word similarity and analogy, and two extrinsic tasks - named entity recognition (NER) and part of speech tagging (POS).

2 Related Works

Learning of word embeddings that capture distributional information has been vital to many NLP tasks. Prediction-based methods such as skip-gram (Mikolov et al., 2013a) and CBOW (Bengio et al., 2003) use neural language modelling for predicting a given word given its context words (or vice-versa) and extract the learned weight vectors as word embeddings. On the other hand, count-based methods derive a co-occurrence matrix of words in the corpus and use matrix factorization techniques like SVD to extract word representations (Levy and Goldberg, 2014). GloVe (Pennington et al., 2014) uses co-occurrence matrix to train word embeddings such that the dot product between any two words is proportional to the log probability of their co-occurrence.

The models that incorporate lexical knowledge

*Equal Contribution

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 115–123
Hyderabad, India, December 2019.

into the word embeddings can be broadly classified into two categories, namely post processing and joint learning. Post processing methods such as (Faruqui et al., 2014; Mrkšić et al., 2016) take the pre-trained word embeddings and modify them by injecting semantic knowledge. The retrofitting method (Faruqui et al., 2014) derives similarity constraints from WordNet and other resources to pull similar words closer together. Whereas, the counterfitting approach, (Mrkšić et al., 2016) also tries to push the antonymous words away from each other. These approaches consider only one-hop neighbours' relations. We improve upon this by considering multi-hop neighbours as well as use structural and information-based similarity scores to determine their relative importance in imposing similarty contraints to the word embeddings.

Joint learning approaches like (Yu and Dredze, 2014; Fried and Duh, 2014; Vashishth et al., 2018) learn word embeddings by jointly optimizing distributional and relational information. For instance, in Yu and Dredze (2014), the objective function consists of both the original skip-gram objective as well as prior knowledge from semantic resources to learn improved lexical semantic embeddings. The recent work by Vashishth et al. (2018) uses Graph Convolutional Networks (GCNs) to learn relations between words and outperforms the previous methods in many language tasks.

Sprinkling: Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis (LSA), learns a distributional representation for words by performing Singular Value Decomposition (SVD) on the term-document matrix. However, the dimensions obtained from LSI are not optimal in a classification setting because it is agnostic to class label information of the training data. The sprinkling method introduced by Chakraborti et al., (2006) improves LSI by appending the class labels as extra features (terms) to the corresponding training documents. When LSI is carried out on this augmented term-document matrix, terms pertaining to the same class are pulled closer to each other. An extension of this method, called adaptive sprinkling (Chakraborti et al., 2007), allows to control the importance of specific class labels by appending them multiple times to the term-document matrix. For instance, in case of double sprinkling, we append the class labels twice to the matrix thus improving the weakly supervised constraints imposed by class labels.

3 Proposed Models

In this section, we discuss the proposed models to incorporate semantic knowledge into word embeddings.

3.1 SS-PPMI & DSS-PPMI

In this approach, we take advantage of Levy and Goldberg's work (2014) in which the authors have shown that the objective function used in Word2vec (Mikolov et al., 2013a) implicitly factorizes a Shifted PPMI (SPPMI) matrix. While there are many methods that attempt to inject semantic knowledge into neural word embeddings, to the best of our knowledge, we have not come across any work that tries to inject semantic knowledge into the SPPMI matrix. In its original form, the SPPMI matrix captures only distributional information. Hence, we are interested in analysing the impact of injecting semantic knowledge into the SPPMI matrix and the effectiveness of the resulting word embeddings.

Inspired from (Chakraborti et al., 2006, 2007), which exploits the class knowledge of the documents by 'sprinkling' label terms into the term-document matrix before matrix factorization, we modify the SPPMI matrix by adding reachability information from lexical knowledge bases such as WordNet and PPDB. In the lexical graphs obtained from these knowledge bases, words are connected by edges representing relations such as synonymy, hypernymy, etc. We say that a word v is reachable from another word u if and only if there exists a path between them in the lexical graph. More formally, let n be the size of the vocabulary. We define the reachability matrix $L_k \in \{0,1\}^{n \times n}$ to be a zero-one square matrix with each element $L_k(u,v)$ indicating if word v is reachable from word u within k hops in the lexical knowledge graph.

We concatenate the reachability matrix with the SPPMI matrix to obtain *Sprinkled Shifted - Positive PMI* (**SS-PPMI**). We then perform SVD on this augmented matrix to obtain the enriched word embeddings.

$$\textit{SPPMI} = max(PMI - \log(neg), 0) \quad (1)$$

$$\textit{SS-PPMI} = \textit{SPPMI} \circ L_k \quad (2)$$

$$\textit{SS-PPMI} \approx U_x \Sigma_x V_x^T \quad (3)$$

$$Embeddings = U_x \Sigma_x^p \quad (4)$$

where $\circ$ denotes the matrix concatenation operation, neg denotes the number of negative samples and x denotes the lower rank approximation of the *SS-PPMI* matrix. *SS-PPMI* matrix is of dimensions $n \times 2n$. Following the work of Levy et al., (2014), we have used p as 0.5 to obtain the word embeddings.

The original motivation for sprinkling technique (Chakraborti et al., 2006) was that documents of same class are brought closer by appending the class labels to term-document matrix. Likewise, words which have strong syntactic relations such as synonymy or antonymy have similar neighbourhood in graphs like WordNet. This translates to these word pairs having similar columns in the reachability matrix. Thus, appending reachability matrix to SPPMI matrix would bring such words closer.

We can further strengthen these constraint by adding the reachability matrix multiple times as done in adaptive sprinkling (Chakraborti et al., 2007). We performed experiments adding reachability matrix twice and we call the resulting matrix as *Doubly Sprinkled Shifted - Positive PMI* (**DSS-PPMI**), which will be of dimensions $n \times 3n$.

3.2 W-Retrofitting

Retrofitting was introduced by Faruqui et al., (2014) and is a method to add semantic information to pre-trained word vectors. The post-processing step modifies the word embeddings such that the embeddings of words with semantic relations between them are pulled towards each other. Formally, given the pre-trained vectors $\hat{Q} = (\hat{q_1}, \hat{q_2} \cdots \hat{q_n})$, and a knowledge base represented by the adjacency matrix A, we need to learn new vectors $Q = (q_1, q_2 \cdots q_n)$ such that following objective $\psi(Q)$ is minimized:

$$\psi(Q) = \sum_{i=1}^{i=n} (\alpha_i \|q_i - \hat{q_i}\|^2 + \sum_{j=1}^{j=n} \beta_{ij} A_{ij} \|q_i - q_j\|^2) \quad (5)$$

The objective is a convex function and we can find the solution using the efficient iterative update method used in Faruqui et al., (2014):

$$q_i = \frac{\sum_{j=1}^{j=n} A_{ij} \beta_{ij} q_j + \alpha_i q_i}{\sum_{j=1}^{j=n} A_{ij} \beta_{ij} + \alpha_i} \quad (6)$$

The β_{ij} term is usually assigned as $degree(i)^{-1}$. This choice of assigning weights can be done in a better way by learning from semantic knowledge sourcea such as WordNet.

Scores	Datasets
Similarity	RG65, WS353S, Simlex-999
Relatedness	WS353R, TR9856
No Distinguishing	MEN, RW, MTunk, WS353

Table 1: The characterization of scores given by different word similarity datasets

We propose a modification to the retrofitting methods called **W-Retrofitting** (weighted retrofitting), where we use WordNet-based similarity scores to obtain a better setting of β_{ij}. For two words w_i and w_j with WordNet similarity score $Sim(i, j)$, β_{ij} is obtained by normalizing the similarity scores across neighbors and is given as: $\beta_{ij} = \frac{Sim(i,j)}{\sum_{j'} Sim(i,j')}$. Since a word can have multiple synsets, the similarity score is the maximum of the similarity scores of all possible pairs of synsets, taking one each from the two words. For information based similarity measures like Lin similarity we compute mutual information from a random subset of Wikipedia corpus containing 100,000 articles. Further, we extend our method to consider nodes which are atmost 2 hops from given node when computing weights.

4 Experimental Setup

4.1 Intrinsic Evaluation

We evaluate the proposed models on word similarity and analogy tasks.

Word similarity: We use MEN (Bruni et al., 2014), MTunk (Radinsky et al., 2011), RG65 (Rubenstein and Goodenough, 1965), Rare Words(RW) (Luong et al., 2013), SimLex999 (Hill et al., 2015), TR9856 (Levy et al., 2015b), WS353 (Finkelstein et al., 2002), WS353S (Similarity), WS353R (Relatedness). Spearman correlation is used as evaluation metric.

Analogy: We evaluated analogy task with Google Analogy (Mikolov et al., 2013a), MSR Analogy (Mikolov et al., 2013b) and Semeval2012 datasets. We follow the standardized setup as explained in (Jastrzebski et al., 2017).

4.2 Sources of Knowledge

We used two sources of semantic knowledge: WordNet (Miller, 1995) and PPDB (Ganitkevitch et al., 2013). We used the same PPDB

knowledge source used in Faruqui et al., (2014). We used WordNet source knowledge from V. Batagelj (2004). The relations considered are synonymy, hypernymy, meronymy and verb entailment. PPDB has 84467 nodes and 169703 edges, WordNet source we used has 82313 nodes and 98678 edges.

We used the latest Wikipedia dump[1] containing 6 Billion wikipedia articles to generate the SPPMI matrix. We followed the same procedure as given in Levy et al., (2015a) and chose the number of negative samples to be default value of 5. In all of our experiments, we chose embedding dimension as 300, which is commonly used in the literature.

4.3 Baselines

We use the following baselines for comparison

1. **GloVe**: Our first baseline is the GloVe embeddings (Pennington et al., 2014) trained on the Wikipedia corpus retrieved from Stanford NLP group website[2].

2. **Retrofit**: We apply the retrofitting technique (Faruqui et al., 2014) on the GloVe embeddings where Wordnet or PPDB was as the source of word relations.

3. **SPPMI**: We perform SVD on the Shifted PPMI matrix (as mentioned in Section 3) without sprinkling.

4. **SynGCN** (Vashishth et al., 2018): This work uses Graph-convolution based methods to impart relational information between words and have shown state-of-art results in many benchmarks. We directly report the available results from the original paper which uses same evaluation benchmarks.

4.4 Extrinsic Evaluation

To further test the effectiveness of the different methods in grounding word meanings, we utilize the embeddings in following tasks. The neural network architectures used for each of the tasks are same as that used in Vashishth et al., (2018).

1. **Part-of-speech tagging (POS):** This task classifies each word of given sentence as one of the part-of-speech tags. We use the LSTM based neural architecture discussed in Reimers and Gurevych (2017) on the Penn treebank dataset (Marcus et al., 1994).

2. **Named-entity recognition (NER):** The goal of this task is to extract and classify named entities in the sentences as person, organisation, location or miscellaneous. We use the model proposed in Lee et al., (2018) on CoNLL-2003 dataset (Sang and Meulder, 2003).

5 Results and Analysis

5.1 SS-PPMI

Reachability Matrix is powerful in capturing semantic information: We proposed a simple sprinkling approach in which a zero-one matrix captures the k-hop reachability information between words in a lexical knowledge graph. In order to see how effectively the reachability matrix captures the lexical knowledge, we performed SVD on the reachability matrix and obtained the word embeddings. Table 2 shows the performance of the obtained embeddings on word similarity task, The dimension of embedding used is 300. Interestingly, we clearly observe that the embeddings obtained from the reachability matrix only (without SPPMI matrix) compete strongly with 300 dimensional pretrained GloVe embeddings on the similarity based datasets. The best performing model gives a Spearman correlation which is **0.19** more than GloVe in Simlex999. Similarly, in RG65 and WS353S, the reachability based embeddings compete well with GloVe. Between the choice of PPDB or WordNet as the lexical knowledge sources, PPDB seems to be more helpful. In general, the performance of reachability-based embeddings increases with increasing the number of hops on the similarity datasets.

In the case of relatedness datasets, the model competes poorly with the baseline-GloVe. This is quite expected as the reachability matrix doesn't capture any information about the word co-occurrence. These observations have been foundational to our proposed *SS-PPMI* and *DSS-PPMI* methods.

SS-PPMI and DSS-PPMI provide significant improvements in word similarity and analogy: Table 3 provides the results with *SS-PPMI* and *DSS-PPMI* approaches on word similarity task with embedding dimension as 300. We clearly observe that the proposed models defeat the baseline

[1] https://dumps.wikimedia.org/enwiki/latest/
[2] https://nlp.stanford.edu/projects/GloVe/

		Similarity			Relatedness		No Distinction			
Lexical Knowledge	Hops - k	SimLex999	WS353S	RG65	WS353R	TR9856	WS353	MEN	MTurk	RW
Baseline - GloVe	-	0.370	**0.665**	**0.769**	**0.560**	**0.575**	**0.601**	**0.737**	**0.633**	**0.411**
PPDB	1	0.507	0.461	0.433	0.127	0.273	0.336	0.284	0.181	0.465
	2	**0.529**	0.567	0.512	0.128	0.261	0.362	0.304	0.261	0.506
WordNet	1	0.077	0.343	0.110	0.151	0.128	0.293	0.161	0.063	0.056
	2	0.209	0.349	0.378	0.163	0.149	0.285	0.275	0.145	0.209

Table 2: Performance of the reachability-based embeddings on similarity datasets. Reported numbers are the Spearman correlation coefficients.

			Similarity			Relatedness		No Distinction			
Method	Lexical Knowledge	hops	SimLex999	WS353S	RG65	WS353R	TR9856	WS353	MEN	MTurk	RW
SPPMI	-	-	0.385	0.728	0.783	0.603	0.625	0.663	0.742	0.599	0.516
SynGCN	-	-	0.455	**0.732**	-	0.457	-	0.601	-	-	0.337
SS-PPMI	PPDB	1	0.386	0.728	0.782	0.604	0.625	0.663	0.742	0.599	0.516
		2	0.398	0.733	0.775	0.619	0.628	0.669	0.743	0.610	**0.521**
DSS-PPMI	PPDB	1	0.386	0.728	0.782	0.604	0.625	0.663	0.742	0.599	0.516
		2	**0.420**	0.733	0.780	0.620	**0.629**	0.668	0.743	0.607	0.528
SS-PPMI	WordNet	1	0.393	0.724	0.792	0.627	0.597	0.667	0.769	0.611	0.464
		2	0.394	0.733	0.793	0.629	0.601	0.671	0.770	0.616	0.435
DSS-PPMI	WordNet	1	0.393	0.724	0.792	0.627	0.597	0.667	0.769	0.611	0.463
		2	0.394	**0.739**	**0.804**	**0.638**	0.599	**0.677**	**0.771**	**0.619**	0.414

Table 3: Results on word similarity datasets using SS-PPMI and DSS-PPMI embeddings

in all the datasets. The margin of improvement is quite high in case of similarity datasets. We see close to **0.21** increase in spearman correlation for Simlex999, **0.04** increase in RG65. This is somewhat expected as we already saw that reachability matrix contains lexical information. Interestingly, we also saw improvements in relatedness datasets where the sprinkling approaches perform narrowly better than SPPMI based approach. In other datasets like WS353, MEN we see improvements of about 0.02 and 0.03 in spearman correlation respectively. Overall, sprinkling significantly improves the performance on word similarity task.

Overall, we observe that Double Sprinkling method (*DSS-PPMI*) works better than *SPPMI* in word similarity task. Increasing the number of hops (k) in the reachability matrix improves the performance in word similarity , in general.

Table 4 shows improvements provided by the sprinkling methods on analogy datasets. We observe marginal improvements over baseline in google and SemEval2012.

5.2 W-Retrofitting

We apply our W-retrofitting model to GloVe (Pennington et al., 2014) embeddings trained on Wikipedia corpus. We experimented with one hop and two hop neighbors and several methods for similarity estimation: inverse path similarity, Jaing-Conrath Similarity (Jiang and Conrath, 1997), Wu -Palmer Similarity (Wu and Palmer, 1994), Leacock-Chowdorov Similarity (Leacock and Chodorow, 1998) and Lin Similarity (Lin et al., 1998). The neighbourhood information for estimating similarity was obtained from either WordNet or PPDB graphs. *We found that Jaing-Conrath Similarity works best for WordNet, inverse path similarity for PPDB*. So, we report results for these similarity measures only.

Method	Graph	hops	Google	SemEval
SPPMI-Baseline	-	-	0.337	0.176
SynGCN			-	**0.234**
SS-PPMI	PPDB	1	0.338	0.175
		2	**0.347**	**0.180**
DSS-PPMI	PPDB	1	0.338	0.176
		2	0.343	0.188
SS-PPMI	WordNet	1	0.122	0.166
		2	0.121	0.165
DSS-PPMI	WordNet	1	0.122	0.166
		2	0.118	0.161

Table 4: Analogy results using proposed SS-PPMI and DSS-PPMI approaches

Word Similarity: The performances of all our models are either comparable or superior to baselines as seen in table 5. We see that using PPDB knowledge source and path based similarity as weights in the retrofit objective functions gives the best performance and outperforms the baselines in most benchmarks.

Analogy: Some of our models outperform retrofitting baselines in Google analogy. In SemEval task, we mostly outperform GloVe but

		Similarity				Relatedness			No Distinction		
Method	Lexical Knowledge	Hops	SimLex999	WS353S	RG65	WS353R	TR9856	MTurk	WS353	MEN	RW
GloVe-baseline		-	0.37	0.665	0.769	0.56	0.575	0.633	0.601	0.737	0.411
SynGCN	-	-	0.455	**0.732**	-	0.457	-	0.601	-	-	0.337
Retrofit-baseline	PPDB	1	0.496	0.7	**0.825**	**0.585**	**0.601**	**0.675**	0.631	0.764	**0.431**
W-retrofit(path)		1	**0.509**	**0.71**	0.824	0.583	0.584	0.669	**0.641**	**0.773**	0.417
		2	0.422	0.628	0.788	0.519	0.525	0.63	0.562	0.722	0.372
Retrofit-baseline	Wordnet	1	**0.434**	0.693	0.774	**0.557**	0.574	**0.642**	0.607	**0.766**	0.387
W-retrofit(jcn)		1	0.432	0.685	0.772	0.543	0.568	0.64	0.6	0.764	0.353
		2	0.399	**0.73**	**0.785**	0.528	**0.579**	0.634	**0.616**	0.764	**0.389**

Table 5: Word Similarity results for W-Retrofitting approach

retrofitting baseline on WordNet gives the best score. The results are summarised in table 6

Similarity	Graph	Hops	Google	SemEval
GloVe		0	**0.717**	0.164
SynGCN			-	**0.234**
Retrofit-baseline	PPDB	1	0.451	0.171
path		1	0.448	0.167
		2	0.248	0.151
Retrofit-baseline	WordNet	1	0.603	**0.184**
jcn		1	**0.701**	0.161
		2	0.693	0.155

Table 6: Analogy results for W-Retrofitting

Model	SimLex999	WS353S	RG65
SPPMI	0.276	0.624	0.671
Retrofitting	0.336	0.624	0.752
W-Retrofitting	0.429	0.656	0.747
Reachability Matrix	0.561	0.567	0.664
Sprinkling	**0.591**	**0.748**	**0.821**
Model	WS353R	TR9856	MTurk
SPPMI	0.509	0.527	0.626
Retrofitting	0.479	0.534	0.623
W-Retrofitting	0.521	0.548	**0.631**
Reachability Matrix	0.194	0.325	0.283
Sprinkling	**0.638**	**0.629**	0.619
Model	WS353	MEN	RW
SPPMI	0.562	0.691	0.359
Retrofitting	0.545	0.708	0.350
W-Retrofitting	0.595	0.726	0.384
Reachability Matrix	0.376	0.325	0.506
Sprinkling	**0.682**	**0.771**	**0.560**

Table 7: Comparison with various baselines for word similarity and relatedness.

5.3 Overall Comparison on Word Similarity

In order to make fair and direct comparison between Sprinkling and Retrofitting, we applied retrofitting and W-retrofitting (using inverse-path similarity over PPDB graph) on the 300 dimensional SPPMI vectors. Table 7 provides the best results of the models on each of the word similarity and analogy datasets. We make the following observations. W-Retrofitting does much better than Retrofitting in similarity datasets, as what we saw with GloVe embeddings. The source of the improvement comes comes from two things: inclusion of two-hop neighbor information and the intelligent choice of weights from WordNet in W-Retrofitting.

Using only the Reachability Matrix provides very good scores in similarity based datasets, but doesn't capture relatedness information at all. Using sprinkling approach, we manage to obtain embeddings that have optimal combination of similarity and relatedness information and this makes it perform better than all the other baselines in similarity, relatedness and analogy tasks.

Method	Graph	Hops	NER	POS
SPPMI-Baseline			82.3	92.9
SS-PPMI	PPDB	1	83.4	93.3
		2	**84.7**	93.4
DSS-PPMI	PPDB	1	82.3	**93.5**
		2	87.3	93.4
SS-PPMI	Wordnet	1	83.5	92.8
		2	83.9	93.2
DSS-PPMI	Wordnet	1	83.2	93.2
		2	83.5	93.1

Table 8: Results on Extrinsic Evaluation tasks using SS-PPMI and DSS-PPMI embeddings

Method	Graph	Hops	NER	POS
GloVe	-		89.1	94.6
SynGCN	-		**89.5**	**95.4**
Retrofit-baseline	PPDB	1	88.8	94.8
path		1	88.7	95
		2	89.2	95.1
Retrofit-baseline	Wordnet	1	88.2	94.5
jcn		1	88.9	95
		2	**89.4**	**95.3**

Table 9: Results on Extrinsic Evaluation tasks using W-Retrofitting

5.4 Evaluation on Extrinsic tasks

The results on extrinsic tasks (discussed in Section 4.4) are given in Tables 8 and 9. In the case of sprinkling methods, we see that there is a clear in-

crease in scores for both the extrinsic tasks from using the proposed SS-PPMI matrix over using only the SPPMI matrix. We also see that models using PPDB perform better. One reason why we do not compare scores of sprinkling based methods with that of GloVe and Retrofitting based ones is that the vocabulary size(number of nodes) in PPDB or Wordnet graphs are lower than that for GloVe. We also didn't consider punctuation symbols in SPPMI unlike GloVe.

In the case of W-retrofitting, scores from the proposed W-Retrofitting model using jcn weights on wordnet graph are very similar to SynGCN model inspite of SynGCN being a more complex model with a lot of hyperparameters. We also see that the other methods of W-retrofitting have comparable performance to SynGCN. We observe improved performance by considering upto 2 hop neighbours over methods considering just 1 hop neighbours. It is quite interesting to see that the proposed light-weight retrofitting model competes strongly with the more complex SynGCN method as shown by the results in Table 9.

6 Conclusion and Future Work

In this paper, we proposed two simple yet powerful approaches to incorporate lexical knowledge into word embeddings. The first approach is a matrix factorization method that 'sprinkles' higher order graph information into the word co-occurrence and we show that it significantly improves the quality of the word embeddings. Second, we proposed a simple modification to the retrofitting method that improves it performance visibly. We showed the improvements of the proposed models over baselines in a variety of word similarity and analogy tasks, and across two popular lexical knowledge bases.

For extrinsic tasks, W-retrofitting showed comparable performance to the state-of-art SynGCN model, (Vashishth et al., 2018) inspite of SynGCN being a more sophisticated model with lots of parameters that constitute the weights of Graph Convolutional layers and linear layers of neural network used as well as many hyperparameters needed for training the neural network (such as number of GCN layers and their dimensions, learning rate, number of epochs, etc.).

In our sprinkling approach, we didn't consider any importance weighting for different relations. One promising direction that can be experimented in future is to use wordnet similarity scores or a combination of co-occurrence and lexical information as importance values in the reachability matrix. We could also use 'adaptive sprinkling' (Chakraborti et al., 2007) to give more importance to relations of specific sets of words.

The more recent methods that achieve the state-of-art results in a variety of language tasks utilize pre-trained models such as Elmo (Peters et al., 2018), BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019). These models that learn context dependent word embeddings are pre-trained for different language tasks and are later fine-tuned for specific tasks. Another direction of research we would like to explore is to study the improvements gained by using our proposed models to initialize the word embeddings before pre-training these models.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Sutanu Chakraborti, Robert Lothian, Nirmalie Wiratunga, and Stuart Watt. 2006. Sprinkling: supervised latent semantic indexing. In *European Conference on Information Retrieval*, pages 510–514. Springer.

Sutanu Chakraborti, Rahman Mukras, Robert Lothian, Nirmalie Wiratunga, Stuart NK Watt, and David J Harper. 2007. Supervised latent semantic indexing using adaptive sprinkling. In *IJCAI*, volume 7, pages 1582–1587.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Daniel Fried and Kevin Duh. 2014. Incorporating both distributional and relational semantics in word representations. *arXiv preprint arXiv:1412.4369*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. 2017. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*.

Jay J Jiang and David W Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

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

Kenton Lee, Luheng He, and Luke S. Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*.

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

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Ran Levy, Liat Ein-Dor, Shay Hummel, Ruty Rinott, and Noam Slonim. 2015b. Tr9856: A multi-word term relatedness benchmark. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 419–424.

Dekang Lin et al. 1998. An information-theoretic definition of similarity. In *Icml*, volume 98, pages 296–304. Citeseer.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *HLT*.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *EMNLP*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *ArXiv*, cs.CL/0306050.

A. Mrvar V. Batagelj. 2004. Wordnet transformed in pajek format, http://vlado.fmf.uni-lj.si/pub/networks/data./dic/wordnet/wordnet.htm, accessed on 10 april, 2019.

Shikhar Vashishth, Manik Bhandari, Prateek Yadav, Piyush Rai, Chiranjib Bhattacharyya, and Partha Pratim Talukdar. 2018. Incorporating syntactic and semantic information in word embeddings using graph convolutional networks. In *ACL*.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *ArXiv*, abs/1906.08237.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 545–550.