DRG2vec: Learning Word Representations from Definition Relational Graph

Xiaobo Shu, Bowen Yu, Zhenyu Zhang, Tingwen Liu*
Institute of Information Engineering, Chinese Academy of Sciences. Beijing, China
School of Cyber Security, University of Chinese Academy of Sciences. Beijing, China
{shuxiaobo, yubowen, zhangzhenyu1996, liutingwen}@iie.ac.cn

***Abstract*—Even with larger and larger text data available, encoding linguistic knowledge into word embeddings directly from corpora is still difficult. The most intuitive way to incorporate knowledge into word embeddings is to use external resources, such as the largest datasource for describing words - natural language dictionaries. However, previous methods usually neglect the recursive nature of dictionaries and cannot capture the high-order relation. In this paper, we present DRG2vec, a novel and efficient approach for learning word representations, which exploits the inherent recursiveness of dictionaries by modeling the whole dictionary as a homogeneous graph based on the co-occurrence of entry and word in the definition. Moreover, a tailor-made sampling strategy is introduced to generate word sequences from the definition relational graph, and then, the generated sequences are fed to the Skip-gram model with semantic negative sampling for word representation learning. Extensive experiments on sixteen benchmark datasets show that leveraging the recursive dictionary graph indeed achieves better performance than other state-of-the-art methods, especially exhibits a weighted average improvement of 8.6% in the word similarity task.**

***Index Terms*—Word embedding, Word similarity, Dictionary, Graph Representation, Text Classification**

I. Introduction

Continuous distributional word representations have become a common technique across a wide variety of natural language processing (NLP) tasks, such as neural machine translation [1, 2] and machine reading comprehension [3, 4]. Most existing embedding models are typically trained based on co-occurrence information in a large corpus. For instance, the training objective of Word2vec [5] is to predict adjacent word(s) given the word or context, i.e., a context window around the target word. This idea aims to capture *semantic relatedness* between words. Unfortunately, *semantic relatedness* is not necessarily equivalent to *semantic similarity* [6]. For example, words such as "expensive" and "cheaper" appear in near-identical contexts, which means that distributional models produce very similar word vectors for such words. Many efforts have been devoted to tackling this issue by introducing external knowledge like WordNet [7, 8, 9] and FrameNet [10, 11]. Despite the great success, these approaches tend to specialize the embeddings to the resource used, while the construction and maintenance of these resources are time-consuming and error-prone.

Recently, Tissier et al. [12] leveraged existing natural language dictionaries to learn word embeddings and proposed a special learning strategy, called Dict2vec. It builds word pairs from dictionary entries as weak supervision to improve the embeddings and exhibits a statistically significant improvement against many competitive solutions. Compared with other external knowledge, dictionary is quite easy to access with fewer errors and is one of the largest refined datasources for describing words. After that, Scheepers et al. [13] proposed a post-process method that tunes pre-trained word embeddings with definitions and lemmas from WordNet. Bosc et al. [14] presented a deep auto-encoder model to generate word definition from word embeddings. However, although achieving convincing performance, according to our empirical study, these methods still suffers from one major drawback: they only focus on the co-occurrences information based on the terms occurring in the definition of a word, and neglect the rich relational structure information in dictionaries. In consideration of the recursive nature of dictionaries - the fact that words inside a definition have their own associated definition, it is intuitive to turn a plain dictionary into a homogeneous word graph. As shown in Figure 1, when viewed individually, the definition of `taxi` expresses that `taxi` is a `car` with a `driver` that you `pay` to take you somewhere. But if we look at `taxi` with a global perspective, this word will have more semantic information as `taxi` is also a `vehicle` and usually running on the `road` and its function is to `carry` something. Such implicit semantic relations in the multi-hop neighbors of the given word will be obscured if we only

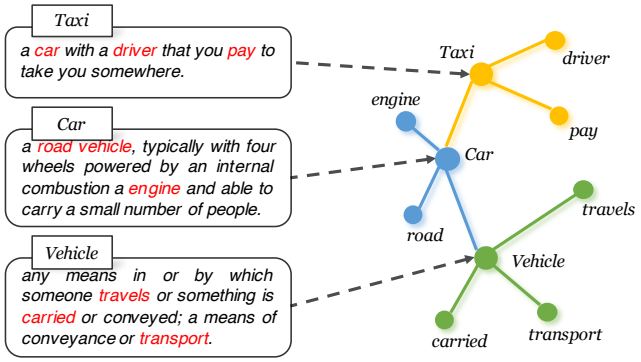

Fig. 1. Toy illustration of the recursive nature of dictionary. The left part is a glance of dictionary and the right part indicate its corresponding graphical representation.

* Corresponding author: Tingwen Liu.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

consider the co-occurrences information of neighborhoods like previous methods.

In this paper, we refer to the relation between one node with its multi-hop neighbors and direct neighbors in the graph as *high-order* and *first-order* relation respectively. Based on the above observation, an obvious question is, how can we leverage the recursive nature of dictionary to inject relations into word representation learning? More importantly, whether word representations learned from the dictionary graph could outperform methods that only utilize plain free text or a few *first order* word pairs (Dict2vec)?

We study these problems in this paper and present a novel word representation learning strategy called DRG2vec based on **D**efinition **R**elational **G**raph (DRG), which is capable of capturing high-order neighborhoods information. Given a dictionary as input, our target is to extract rich semantic relation of words by making full use of the definition resource. For this purpose, DRG2vec first builds an undirected labeled weighted graph where vertices represent words, and the edge between two word nodes is assigned a specific weight value, according to the importance of one node over another, we adopt TF-IDF value as it. Next, to effectively and fully exploit the recursive nature of the definitional relation graph, a tailor-made sampling strategy is proposed for generating word sequences from DRG with random walk, which balances the effect of breadth-first-search and deep-first-search algorithm. Finally, the generated word sequences from DRG and the free-text from Wikipedia are fed to the Skip-gram model for producing fixed-length continuous-valued word representations.

The experimental results demonstrate that DRG2vec achieves significant improvements on twelve word similarity datasets, and meanwhile is comparable with the state-of-the-art methods on four text classification datasets. Further analysis indicates that DRG2vec can fully exploit the recursive structure of dictionaries with the reasonable definition relational graph.

II. METHOD

In this section, we detail our DRG2vec approach. Section II-A introduces the way to build definition relational graph (DRG). Section II-B introduces a tailor-made sampling strategy to generate sequence based on DRG. In Section II-C, we review the objective function of Skip-gram, then propose a novel negative sampling strategy.

A. Graph Construction

In this section, as depicted in Figure 2, we aim to build a large definition relational graph. Thus, the global word co-occurrence and definition relation can be modeled explicitly, which is convenient to introduce the following graph method.

Our design philosophy is motivated by the following observation: words inside a definition have their associated definition. In particular, `vehicle` is in the definition of `car`, and `vehicle` is explained by `carried`, there is no doubt that `carried` has some relevance to the defined word `car`. Furthermore, not all the words in the definition contribute equally to the interpretation, a word needs only a few keywords to interpret. As in the definition of `vehicle`, words like `which`, `means` and `something` are far less effective in interpretation than `travel`, `carried` and `transport`. Based on the above observation, to represent the semantic relevance between words, a dictionary is transformed into an undirected semantic graph. Formally, consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}_{i=1}^{i=n}$ and $\mathcal{E} \subseteq \{e_{ij}\}_{i,j=1}^{i,j=n,i\neq j}$ are sets of nodes and edges, respectively. Every node is assumed to be a word and if v_j appears in the definition of v_i, an edge $e_{ij} \in \mathcal{E}$ will be established to connect these two words. By doing this, words in the graph will connect directly with not only their definition words, but also the words they interpret.

Up to now, the weights of edges in the graph are all the same, but as mentioned above, different words have different importance in a definition, that is, the more important a word is, the higher its weight should be, and vice versa. To model such information, we introduce the TF-IDF algorithm to distinguish the important words. The weight of edge between v_i and v_j is defined as the *tf-idf* value of word pair (v_i, v_j), which is computed as:

$$\textit{tf-idf}(v_i, v_j) = \textit{tf}(v_i, v_j) \times \textit{idf}(v_j) \tag{1}$$

$$\textit{tf}(v_i, v_j) = \frac{|N_{ij}|}{\sum_{v_k \in N_i} |N_{ik}|} \tag{2}$$

$$\textit{idf}(v_j) = \log\left(\frac{|N|}{\sum_{i=0}^{|N|} sgn(|N_{ij}|)}\right) \tag{3}$$

where N is the set which contains all of the definitions, N_i is the definition of word v_i, and N_{ij} represents word v_j in the definition of word v_i. u_{ij} is the final weight on edge e_{ij}, sgn denotes the sign function. If two words define each other, we add *tf-idf*(v_i, v_j) and *tf-idf*(v_j, v_i) together. The final weight is defined as:

$$u_{ij} = u_{ji} = \textit{tf-idf}(v_i, v_j) + \textit{tf-idf}(v_j, v_i) \tag{4}$$

Note that there is no self-loop in the graph, because it makes no sense to consider the semantic relevance of a word to itself. Besides, we have also tried multiple alternative methods for weight calculation, such as Pointwise Mutual Information (PMI) and Term Frequency (TF), some preliminary experimental results show that *tf-idf* always works better than other methods.

B. Sequence Generation

After building the dictionary graph, we aim to learn the latent semantic embeddings for each node (i.e., each word) in $\mathcal{G}$. In order to obtain sequence set that can be directly used to learn word representations, we first generate paths from $\mathcal{G}$ by random walks [15], and each generated path is regarded as a sentence by treating each node in the graph as a word.

Formally, let n_x denote the x-th node in the generated sequence, given a source node n_0, the random walk algorithm is employed to generate a node sequence from the graph, and

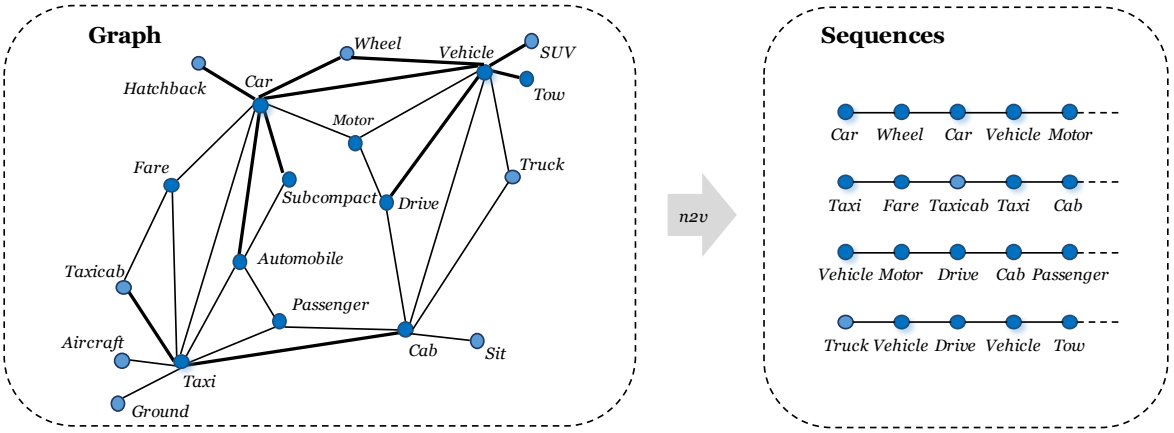

Fig. 2. Schematic diagram of the definition relational graph (DRG) and sequence generation. For good visualization, we choose 6 words with the highest weight which directly connected to Car, Taxi or Vehicle (plurals and other tenses of words are filtered out). Different thicknesses of edges in the graph mean different weights between nodes.

the probability of sampling the node n_{x+1} based on n_x is defined as follows:

$$p(n_{x+1} = v_j | n_x = v_i) = \begin{cases} \pi_{ij}/Z, & \text{if } e_{ij} \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $Z = \sum_j \pi_{ij}$ is the normalization constant for node v_i. By default, $\pi_{ij} = u_{ij}$. Specifically, the random walk algorithm first starts from an initial node v_s, then randomly select an edge directly connected to v_s on the basis of edge weights, and reach a neighbor node v_n. Next, v_n will be regarded as a new source node v'_s and the random walk process will restart all over again, until the iteration is end. By recording all the nodes in the process of traversal, we can obtain a sequence of word nodes for training word embeddings.

However, such a random walk strategy based on the above probability function is approximate to the Depth-First Sampling (DFS) algorithm, because they all tend to traverse away from the source node, which will lead to the generated sequences contain more semantic relatedness. On the contrary, we believe that Breadth-First Sampling (BFS) algorithm can restrict the travel range around the source node, so that the semantic similarity among words will be more concerned in the sequences. Considering that DFS can explore larger parts of the graph, while BFS plays a key role in characterizing the local neighborhoods accurately, we follow the idea of node2vec [16] to balance DFS and BFS and take place of these algorithms to generate sequence. In fact, node2vec is an extension of random walk which better controls the depth-first and breadth-first property of random walks compared with other variations.

Specifically, instead of directly setting $\pi_{ij} = u_{ij}$ in Equation 5, node2vec utilizes a biased transition probability as the replacement to control how fast the walk explores and leaves the neighborhood of the source node. Formally, consider a walk that just traversed edge e_{hi} from v_h and now resides at node v_i, it needs to decide on the next step so it evaluates the transition probabilities π_{ij} on edges e_{ij}. We set the unnormalized transition probability as follows:

$$\pi_{ij} = \alpha(h, j) \cdot u_{ij} \tag{6}$$

$$\alpha(h, j) = \begin{cases} p^{-1}, & \text{if } d(h, j) = 0 \\ 1, & \text{if } d(h, j) = 1 \\ q^{-1}, & \text{if } d(h, j) = 2 \end{cases} \tag{7}$$

where $d(h, j)$ is the shortest path distance between node v_h and v_j. Intuitively, parameters p and q allow our sampling procedure (approximately) interpolate between BFS and DFS and thereby reflect an affinity for different notions of node equivalences [16]. Here, p is named *return parameter* and q named *in-out parameter*, a high value of p ($p > 1$) ensures that the walk is less likely to sample a visited node in the following two steps (unless the next node in the walk had no other neighbors), while $q < 1$ will make random walk step far away from the source node.

C. Learning Word Representations

We directly utilize the widely-used model Skip-gram to learn word representations from our generated paths, because Skip-gram has well-balanced effectiveness as well as efficiency [17]. Our objective function is derived from the noise contrastive estimation, which is a more efficient objective function than the log-likelihood function:

$$L = -\frac{1}{n} \sum_{t=1}^{n} \sum_{k \neq 0}^{|k| \leq C} \psi^{+}_{w_{t,t+k}} + \sum_{w_i \in \mathcal{N}(w_t)} \psi^{-}_{w_t, w_i} \tag{8}$$

Here, w_t is the target word, w_{t+k} is one of the context words within window size C. $\psi^{+}_{w_{t,t+k}} = \log \sigma(\mathbf{v_t} \cdot \mathbf{v_{t+k}})$, $\psi^{-}_{w_t, w_i} = \log \sigma(-\mathbf{v_t} \cdot \mathbf{v_j})$, where $\sigma(x)$ is the sigmoid function, and $\mathbf{v_t}$ is the embedding of word w_t. $\mathcal{N}(w_t)$ is a set that consists of the sampled noisy context words for the word w_t. For each word

w_t, standard negative sampling strategy generates $\mathcal{N}(w_t)$ by randomly selecting k words from the vocabulary:

$$\mathcal{N}(w_t) = \{w_i\}^k, \quad w_i \in \mathcal{V}_w \backslash \{w_t\} \tag{9}$$

However, such a random sampling strategy does not consider semantic relations among words, and the probability that w_i and w_t are related is not necessarily zero. With the semantic path we have generated above, it is possible to better ensure that this is less likely to occur. More specifically, for a given word w_t and the sampled word sequences, we prohibit a negative example of w_t to be a word that co-occurs with w_t in the word sequences. Finally, the Equation 9 can be modified to the following form:

$$\mathcal{N}(w_t) = \{w_i\}^k, \quad w_i \in \mathcal{V}_w \backslash \{w_t, \mathcal{P}(w_t)\} \tag{10}$$

where $\mathcal{P}(w_t)$ is the word set containing all the words showing up in the same sequence with w_t, and $\mathcal{P}(w_t)$ is set to $\emptyset$ when w_t does not belong to DRG. For better identification, we name such a strategy as semantic negative sampling.

III. EXPERIMENTS

This section first introduces the experimental settings, and then presents the performance comparison results with baseline methods as well as case studies to validate the effectiveness of DRG2vec and the superiority of DRG.

A. Experimental Settings

1) Training Data: The training data we use includes Wikipedia dump and online Dictionary Resources. The details are as follows:

- **Wikipedia**. We utilize the English dump from Wikipedia updated on March 1, 2019[1]. We run the script offered by Mahoney[2] to remove non-English words and special symbols, convert uppercase letters to lowercase, and finally we achieve the corpus containing 4.6 billion tokens.
- **Dictionary Resources**. The online dictionary source we use is the same as Dict2vec[3] [12], which includes English version of Cambridge, Oxford, Collins and dictionary.com. For each word, we download its definitions in the above four dictionaries and extract such definitions from HTML pages with regex. After removing stop words and punctuation, we concatenate all definitions of each word. Overall, the Wikipedia dump contains 2.5M unique words, of which about 150K words' definition can be fetched from above online dictionary resources.

2) Implement Details: We follow the same evaluation protocol as Word2vec and fastText to provide the fairest comparison against competitors, we use 5 negative samples, 5 epochs, a window size of 5 and a vector size of 100. The parameters in Dict2vec are the default setting by original paper ($N_w = 4, N_s = 3, \beta_s = 0.8, \beta_w = 0.45$). For our method, the times and depth of random walk are both set to 40. To examine how the different choices of parameters affect the performance, we compare the all round performances of three different settings: $p = 10.0$, $q = 1.0$, $p = 1.0$, $q = 10.0$ and $p = 1.5$, $q = 5.0$, which we refer to as DFS-like (DRG2vec-d), BFS-like (DRG2vec-b) and Trade-off (DRG2vec) respectively. We run 3 times for each experiment then report the average results.

3) Baselines: To compare and evaluate our method comprehensively, we employ several models as baselines, which can be divided into two genres. (1) Context-based genre, we train **Word2vec** [17] and **fastText** [18] on the same corpus described in the section of training data with the same hyper-parameters [4]. For the sake of fairness, we train Word2vec with the Skip-gram model. (2) Dictionary-based genre: **Dict2vec** [12] leverages word pairs extracted from the definitions weighted differently with respect to the strength of the pairs to learn word embeddings; **CPAE** [14] utilizes auto-encoder model and soft weighting scheme to bring the input embeddings closer to the encoded definition embeddings. Besides, following previous studies [10, 12], we also retrofit the learned embeddings with Faruqui's method to compare another method using additional resources. The external knowledge we used here contains the WordNet semantic lexicon [7], pairs extracted from Dict2vec and our DRG.

4) Evaluation Tasks: Following previous work [12], we evaluate our method on two kinds of tasks with 16 public datasets:

- **Word Similarity**: We follow the standard method for word similarity evaluation by computing the Spearman's rank correlation coefficient between human similarity evaluation of pairs of words, and the cosine similarity of the corresponding word vectors. More specifically, the closer the score is to 1, the closer the embedding is to human judgment. To improve readability, we multiply the score by 1000. To evaluate the performance of Word Similarity, we use MC-30 [20], RG-65 [21], WS-353-ALL [22], MEN-3K [23], MT-287 [24], MT-771 [25], RW-STAN [26], SimVerb-3500 [27], SimLex-999 [28] and YP-130 [29] classic datasets. Furthermore, WS-353-ALL is also further divided into two fragments which describe word similarity (WS-353-SIM) and relatedness (WS-353-REL) respectively. We also compute the average results by weighting each score by the number of pairs evaluated in its dataset in the same way as [30].
- **Text classification**: Our text classification task includes four datasets: AG-News [5], Yelp reviews (i.e., Yelp Pol. and Yelp Full)[6], and DBpedia [31]. We make evaluation by training a neural network composed of a single hidden layer where the input layer corresponds to the bag of words of a document, and the output layer is the probability to belong to each label. The weights between

[1]https://dumps.wikimedia.org/enwiki/20190301/

[2]http://mattmahoney.net/dc/textdata\#appendixa

[3]https://github.com/tca19/dict2vec

[4]We have also tried GloVe [19] with the official implementation on our training data, but the results are lower than all other baselines (weighted average score 40.7%), thus we do not report GloVe's results in the experiment.

[5]https://www.di.unipi.it/œgulli/AG_corpus_of_news_articles.html

[6]https://www.yelp.com/dataset_challenge

the input and the hidden layer are initialized with the generated embeddings and fixed during training, so that the evaluation score solely depends on the embedding. We update the weights of the neural network classifier with gradient descent. We use the same training data and test data in all approaches.

B. Experimental Results

1) Word Similarity: Table I (top) reports the Spearman's rank correlation scores of all comparison models. From the results, the first conclusion we draw is that our DRG2vec method outperforms all baseline methods in terms of the weighted average score of 12 word similarity datasets (+25.8% over Word2vec, +24.6% over fastText, +10.2% over Dict2vec, +8.6% over CPAE). We attribute such substantial performance gain to two design choices: (1) learning word embeddings through definition relational graph indeed makes full use of the structural and semantic information of dictionary, and could captures high-order neighborhoods information; (2) Our sequence generation strategy is quite effective to capture the *semantics similarity*. Secondly, we notice that dictionary-based methods (Dict2vec, CAPE and our DRG2vec) outperform context-based methods (Word2vec, fastText) marginally, which can be intuitively explained that word definition provides extra signals to boost the performance of word embeddings. When compared with other dictionary-based baselines, our DRG2vec method achieves significant improvements on several datasets, which is a strong evidence that learning word representation base on DRG can capture richer semantic information compare to other "plain" methods. Finally, we also note that the performances of BFS-like and DFS-like strategies are comparable. Specifically, DRG2vec-d outperforms DRG2vec-b on the WS-353-REL dataset (for word relatedness) while DRG2vec-b surpasses DRG2vec-d on the WS-353-SIM dataset (for word similarity), which demonstrates that DFS-like strategy learns more about semantic relatedness information while BFS-like strategy encourages the sequences to pay more attention to similar words. DRG2vec model actually makes a trade off between DRG2vec-b and DRG2vec-d, thus achieving better performance.

Beyond that, the Faruqui's retrofitting method improves the word similarity scores on all methods for most datasets (see Table II). It is worth mentioning that when the baseline methods are retrofitted, their scores are still worse than our non-retrofitted model (every percentage on the "v.s. our" line are negative), which means that using our semantic graph is a sensible way to improve the quality of the embeddings. We also notice that the weighted average score of our model is slight increased by retrofitting method with WordNet, which can be explained that DRG2vec has captured both first-order and high-order definitional relations, and the addition of synonyms knowledge can further improve word embeddings.

2) Text Classification: In Table I (bottom), we report the classification accuracy for these considered datasets. From the results, we can see that DRG2vec achieves better results on AG-News, DBPedia and Yelp Full datasets than baselines, and the performance on Yelp Pol. is very close to that of state-of-the-art approach. It suggests that incorporating external word definition knowledge into word embedding by our approach not only improves intrinsic task, but also benefits extrinsic tasks, which is consistent with the conclusion of Dict2vec.

C. Effectiveness Analysis

1) Effectiveness of Definition Relational Graph: To further validate the effective of our constructed definition relational graph, we extract word pairs from with WordNet, Dict2vec and DRG[7]. In Table II, we report the relative percentage changes of Spearman's rank correlation score after the Faruqui's retrofitting method applied to pre-trained word embeddings (Word2vec and fastText) with different sources of word pairs. Compared with WordNet and Dict2vec, pairs extracted from our DRG improve all three baselines in term of the weighted average score, which shows that DRG is more effective than Dict2vec and human-handed WordNet resources in some way. Notice that when applying retrofitting to our trained word embeddings (DRG2vec), only WordNet could bring some improvement. It proves that some relational information in the WordNet lexicon may be complementary to our definitional relation.

2) Effectiveness of High-order: To demonstrate the influence of high-order relation and the effectiveness of our graph-based approach for capturing high-order relation between words, we extract word pairs from sequences generated by random walk, in which the order is defined as the number of steps far away from the source word, and then use them to retrofit different kinds of word embeddings.

From Figure 3, we can observe that the performances of all comparison models except ours generally display a trend of rising first and then decreasing after second-order, which indicates that the first two order words (neighborhoods) share the most semantic relevance with the central word, and higher-order relation may suffer from the issue of semantic drift. Nevertheless, when the number of order is less than 5, the high-order relation can still boost the performance compared with original words representation, which again confirms the effectiveness and rationality of our random walk strategy to capture diverse order semantic relations. One may notice that the high-order relation can not help trained DRG2vec embeddings make further progress. The reason behind such phenomenon is still under investigation, we presume that our approach has caught the high-order relation of words to some extent.

3) Effectiveness of Semantic Negative Sampling: In order to explore the effectiveness of our semantic negative sampling strategy, we replace the sampling strategy used in Dict2vec and Word2vec with ours and re-implement their results on the 12 word similarity datasets. Experimental results indicate that our sampling strategy can significantly improve the average score of word similarity when using the same model(+1.9% over

[7]We extract the three highest weight neighborhoods for every word as word pairs. WordNet: https://github.com/mfaruqui/retrofitting, Dict2vec: https://github.com/tca19/dict2vec, DRG: https://github.com/shuxiaobo/DRG2vec.

TABLE I

SPEARMAN'S RANK CORRELATION COEFFICIENTS BETWEEN VECTORS' COSINE SIMILARITY AND HUMAN JUDGEMENT FOR SEVERAL DATASETS (TOP) AND ACCURACIES ON TEXT CLASSIFICATION TASK (BOTTOM). WE TRAIN AND EVALUATE EACH MODEL 3 TIMES AND REPORT THE AVERAGE SCORE FOR EACH DATASET, AS WELL AS THE WEIGHTED AVERAGE FOR ALL WORD SIMILARITY DATASETS.

	W2V	FT	D2V	CPAE	DRG2vec-d	DRG2vec-b	DRG2vec
MC-30	786	725	861	858	859	864	**876**
MEN-3K	699	659	746	705	**805**	801	804
MT-287	628	600	648	622	700	702	**703**
MT-771	583	579	675	615	755	**763**	759
RG-65	704	640	860	806	888	881	**895**
RW-STAN	343	432	505	483	430	405	**464**
SimLex-999	318	321	452	518	564	563	**569**
SimVerb-3500	209	248	417	478	572	570	**574**
WS-353-ALL	663	587	725	653	768	768	**768**
WS-353-REL	619	519	637	562	**690**	684	683
WS-353-SIM	733	693	741	726	810	817	**819**
YP-130	411	509	635	723	750	747	**753**
W.average	510	522	666	682	760	754	**768**
AG-News	831	821	816	810	830	821	**834**
DBPedia	913	916	924	927	930	910	**933**
Yelp Pol.	802	**818**	814	813	813	811	812
Yelp Full	431	449	452	436	451	430	**455**

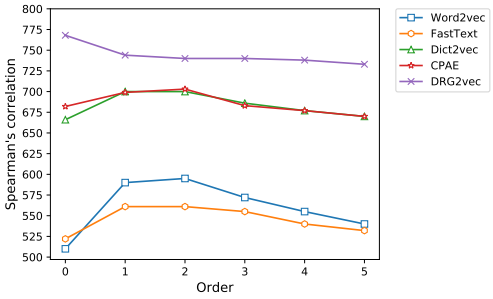

Fig. 3. The weighted average scores of word similarity task after retrofitting pre-trained word embedding by different order word pairs.

Word2vec and +1.3% over Dict2vec, in which the sequence length walked from graph is set to 40), which suggests that the negative samples we extract from DRG are more useful on general approaches. Meanwhile, it also indicates that our negative sampling method can take full account of the semantic information among words, so that the model is endowed with the ability to differentiate unrelated words rather than simply putting the related words closer.

D. Case Study

In Table III, we present some cases of the most similar word to intuitively illustrate the characteristics of word embeddings produced by our approach. In the first case, `coffee` is a noun describing things containing coffee beans, it is shown to be more similar with `cocoa` and `lattes` than `milk` or some else words. In the second case, `good` should be closer to `excellent`, `better` or `best` in the semantic space, negtive words like `bad` and `poor` are antonyms of `good`, and there should not be much similarity between them. Overall, our method can learn the real *semantic similarity* from definition, and it confirms that that using refined dictionary knowledge can actually help to get the real sense of the words.

IV. RELATED WORK

Word Representation. Recently, the context-free pre-trained word embeddings have been used in kinds of NLP tasks as the distributed representations of words [32]. The Word2vec [5, 17], likes Skip-gram (SG) and Continuous-Bag-Of-Words (CBOW) models, gains further popularity thanks to its simplicity and effectiveness. Both of them iterate overall *(target, context)* pairs from every window of the corpus and tries to predict target word(s) by given word(s). the difference is that Skip-gram model predicts the neighborhoods for a given word, while CBOW predicts the target word using the context. Considering the great success of Word2vec, there are many follow-up works based on them. FastText [18] uses a training algorithm based on Skip-gram, it represents each word as a bag of character n-grams to capture more semantics and solve out-of-vocabulary (OOV) problem at the same time. To capture global information in the corpus, GloVe [19] constructs a global statistical co-occurrence matrix, and learns word embedding by factorizing the matrix.

However, all the methods we mentioned above just modeling the co-occurrence information into word representation. With larger and larger text data available on the Internet, more works concentrate on extracting and encoding linguistic knowledge into word embeddings directly from external resources [9, 14, 33]. Approaches in this area can be roughly divided into two categories: pre-training and post-processing. (1) The pre-training methods used to inject external knowledge by modifying loss function or text corpus. For example, [9] include prior knowledge about synonyms from WordNet and

TABLE II

PERCENTAGE CHANGES OF WORD SIMILARITY SCORES FOR SEVERAL DATASETS AFTER THE FARUQUI'S RETROFITTING METHOD IS APPLIED. WE COMPARE EACH MODEL TO THEIR OWN NON-RETROFITTED VERSION (VS SELF) AND OUR NON- RETROFITTED VERSION (VS OUR). A POSITIVE PERCENTAGE INDICATES THE LEVEL OF IMPROVEMENT OF THE RETROFITTING APPROACH, WHILE A NEGATIVE PERCENTAGE SHOWS THAT THE COMPARED METHOD IS BETTER WITHOUT RETROFITTING.

		Word Pairs of WordNet				Word Pairs of Dict2vec				Word Pairs of DRG			
		FT	D2V	CPAE	Drg2vec	FT	D2V	CPAE	Drg2vec	FT	D2V	CPAE	Drg2vec
MC-30	vs.self	+17.6	+1.5	+5.2	+0.6	+3.4	+4.1	+0.1	+2.7	+11.4	+0.7	+0.9	+0.5
	vs.our	**-2.7**	**-0.3**	+3.0	-	**-14.4**	+0.6	**-2.0**	-	**-7.8**	**-2.6**	**-0.6**	-
MEN-3k	vs.self	-2.2	-2.4	-3.8	-0.6	-15.4	-0.1	-2.69	-4.3	+5.4	+3.4	+0.4	-0.4
	vs.our	**-20.0**	**-9.5**	**-15.6**	-	**-30.8**	**-9.9**	**-14.6**	-	**-13.6**	**-6.7**	**-6.4**	-
MT-287	vs.self	-10.8	+3.0	-3.5	-2.3	-13.8	+3.7	+9.00	-1.5	-0.3	+5.9	+7.3	-1.6
	vs.our	**-25.2**	**-6.7**	**-14.6**	-	**-27.7**	**-4.4**	**-3.5**	-	**-16.4**	**-2.5**	+1.5	-
MT-771	vs.self	+6.0	+1.7	-1.9	+1.0	-16.7	+2.1	-2.43	-1.1	+6.7	+4.2	+4.3	-0.6
	vs.our	**-19.4**	**-9.8**	**-20.5**	-	**-36.7**	**-8.2**	**-20.9**	-	**-18.8**	**-6.4**	**-7.6**	-
RG-65	vs.self	+9.0	-0.2	+2.8	+0.5	-2.8	+4.5	-4.71	-4.8	+6.0	+2.6	+4.2	+0.2
	vs.our	**-6.9**	**-4.1**	**-7.3**	-	**-17.0**	**-2.6**	**-14.1**	-	**-9.4**	**-4.4**	**-4.2**	-
RW-STAN	vs.self	-28.2	-20.0	-31.4	-15.0	-7.8	-19.8	-5.17	-11.8	+1.3	+7.9	+7.2	-0.7
	vs.our	**-41.8**	**-24.2**	**-28.6**	-	**-25.3**	**-24.3**	**-1.2**	-	**-17.8**	+1.8	+13.3	-
SemLex	vs.self	+24.9	+9.5	+2.8	+8.8	+8.0	+12.9	-5.01	-6.6	+11.5	+5.1	+5.2	-1.5
	vs.our	**-30.3**	**-14.0**	**-6.3**	-	**-39.7**	**-9.2**	**-13.5**	-	**-37.8**	**-15.4**	**-14.4**	-
SimVerb	vs.self	+38.7	+5.2	+6.0	+6.5	+34.2	+18.7	-1.29	-12.1	+27.0	+12.0	+4.6	-1.8
	vs.our	**-41.0**	**-4.8**	**-11.6**	-	**-42.9**	**-18.8**	**-26.1**	-	**-46.0**	**-23.4**	**-31.3**	-
WS-ALL	vs.self	-15.8	-1.1	-12.2	-4.6	-24.3	-2.8	-1.94	-8.0	+3.5	+1.6	-1.2	-0.9
	vs.our	**-36.1**	**-7.3**	**-25.3**	-	**-42.6**	**-6.7**	**-25.1**	-	**-21.4**	**-2.4**	**-8.5**	-
WS-REL	vs.self	-38.5	-3.2	-21.7	-7.5	-40.4	-7.7	-6.37	-8.9	-0.5	+1.7	-2.3	-0.2
	vs.our	**-53.8**	**-10.8**	**-35.5**	-	**-55.2**	**-10.5**	**-31.1**	-	**-25.3**	**-1.3**	**-13.3**	-
WS-SIM	vs.self	+2.5	+7.4	-3.4	-0.8	-12.5	+5.6	-7.30	-3.9	+4.6	+3.2	+1.3	+0.6
	vs.our	**-13.8**	**-3.5**	**-14.4**	-	**-26.5**	**-2.3**	**-17.8**	-	**-12.1**	**-4.4**	**-0.1**	-
YP-130	vs.self	+14.5	+4.0	+0	+3.5	+23.5	+8.5	-5.94	-5.1	+8.0	+0.6	-0.4	-0.5
	vs.our	**-24.3**	**-14.2**	**-3.9**	-	**-18.4**	**-11.2**	**-9.6**	-	**-28.6**	**-17.7**	**-11.1**	-
W.average	vs.self	+1.3	+0.5	-4.1	+1.6	-4.2	-4.8	-5.71	-7.4	+8.3	+6.3	+3.1	-1.0
	vs.our	**-30.6**	**-15.8**	**-14.2**	-	**-34.9**	**-12.1**	**-15.6**	-	**-25.7**	**-10.2**	**-10.3**	-

the Paraphrase Database in a joint model built upon Word2vec. Prop [34] first construct a relational graph from text corpora by lexical patterns, then minimizes the difference between the predicted and the actual co-occurrences. Kiela et al. [35] adds new contexts from an external thesaurus or a norm association base in the Skip-gram loss function for optimization, so that it can specialize the embedding on kinds of relation. (2) The postprocessing methods improve word embedding by fine-tuning already trained word embedding with lexical databases. Faruqui et al. [6, 10] introduce a retrofitting method in which they postprocess learned vectors with respect to semantic relationships extracted from additional lexical resources. Vulić et al. [36] inject external linguistic constraints into the initial vector space, and the related word pairs are more close in the transformed Euclidean space, so as to emphasize the asymmetric relation of lexical entailment. SemGCN [8] builds a word graph and represent each node (or word) by graph convolution network, then uses the neighborhoods to predict the center word.

Finally, we turn to the works most relevant to our model, which also involves dictionaries and word embeddings. Dict2vec [12] combines the Skip-gram objective with a cost for predicting related words, in which the related words either form strong or weak pairs with the target word in the definition. CAPE [14] introduces the soft weight-tying scheme into auto-encoder model and learns word embeddings by processing dictionary definitions and trying to reconstruct them. In spite of a similar dictionary resource shared by these studies, our work is unique in that we find and further make full use of the recursive nature of dictionary, then model the whole dictionary as a homogeneous graph to characterize the neighborhoods accurately.

Network Representation. In the network representation field, conventional paradigm of node feature generation is typically based on feature extraction techniques, which involve some hand-crafted seed features based on network properties [37, 38]. After that, dimensionality reduction techniques (e.g., PCA, IsoMap) are proposed to reduce the cost of extracting

TABLE III
CASE STUDY FOR QUALITATIVE ANALYSIS. GIVEN THE TARGET WORD, WE LIST THE TOP 8 SIMILAR WORDS FROM EACH ALGORITHM SO AS TO OBSERVE THE DIFFERENCES.

	Word2vec		Dict2vec		CPAE		DRG2vec	
coffee	tea	sugar	cocoa	tea	java	bean	cocoa	coffees
	cotton	cocoa	molasses	arabica	coffea	cacao	coffea	tea
	rice	vegetables	teas	flour	chocolate	carob	lattes	decaffeinated
	meat	milk	oatmeal	leche	coffeepot	cocoa	chocolate	cuppers
good	bad	poor	better	decent	better	asset	excellent	better
	necessary	true	pretty	luck	best	upstanding	mediocre	best
	excellent	better	excellent	beneficent	goodness	worthless	marvelously	gentilesse
	therefore	clear	bad	really	wholesomeness	virtue	reputable	unaesthetic
bad	good	poor	badness	wrong	worse	innocuous	appalling	badness
	luck	wrong	worse	ill	unsound	evil	misfortunes	jinx
	terrible	better	dooms	execrable	unfit	noxious	abominable	misfortunes
	happen	worse	wickedly	horrible	badly	badness	appallingly	unforgivable

features from networks. However, these methods still suffer from both computational and statistical performance drawbacks. In addition, these methods optimize for objectives that are not robust to the diverse patterns observed in networks (such as homophily and structural equivalence) and make assumptions about the relationship between the underlying network structure and the prediction task. Recently, Inspired by the Skip-gram model [17], Deepwalk [39] and LINE [40] established an analogy for networks by representing a network as a "document". in which words in documents are represented as nodes and the random walk strategy is utilized to sampling ordered word sequences from the network. However, there are many possible sampling strategies for nodes, resulting in different learned feature representations. In fact, the key to win in kinds of tasks across all networks is adapt a better sampling strategy. Node2vec [16] designed a flexible objective that is not tied to a particular sampling strategy and provides parameters to tune the explored sampling space, so that it could balance the deep-first sampling and breadth-first sampling by random walk strategy. In this paper, we employ the node representation approaches to help us mine the structure information of our definition relational graph (DRG).

V. CONCLUSIONS AND FUTURE WORK

In this paper, we presented DRG2vec, a novel graph approach for learning word embeddings using lexical dictionaries. Dictionaries are transformed into a word semantic graph based on the co-occurrence of entry and word in the definition. Then a deep random walk strategy is introduced to generate sequences from the graph, which are then fed to the Skip-gram model for word representation learning. Experimental results show that DRG2vec significantly outperforms the state-of-the-art solutions. They also demonstrate that nature language dictionary contains more semantic information than handmade resources in some ways. In the future, we will conduct research in: (1) how to utilize dictionaries to learn multi-sense word embedding as dictionary defines every sense of word, (2) exploring more advanced graph representation methods to improve the performance of word embedding.

VI. ACKNOWLEDGEMENTS

This work was supported by the National Key Research and Development Program of China (No.2016YFB0801003) and the Strategic Priority Research Program of Chinese Academy of Sciences (No.XDC02040400).

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. of NeurIPS*, 2014.

[3] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," in *Proc. of ACL*, 2017.

[4] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," *arXiv preprint arXiv:1705.00106*, 2017.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[6] N. Mrkšic, D. OSéaghdha, B. Thomson, M. Gašic, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Counter-fitting word vectors to linguistic constraints," in *Proc. of NAACL*, 2016.

[7] G. A. Miller, "Wordnet: a lexical database for english," *CACM*, 1995.

[8] S. Vashishth, M. Bhandari, P. Yadav, P. Rai, C. Bhattacharyya, and P. Talukdar, "Incorporating syntactic and semantic information in word embeddings using graph convolutional networks," in *Proc. of ACL*, 2019.

[9] M. Yu and M. Dredze, "Improving lexical embeddings with semantic knowledge," in *Proc. of ACL*, 2014.

[10] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," in *Proc. of NAACL*, 2015.

[11] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "Ppdb: The paraphrase database," in *Proc. of NAACL*, 2013.

[12] J. Tissier, C. Gravier, and A. Habrard, "Dict2vec: Learning word embeddings using lexical dictionaries," in *Proc. of EMNLP*, 2017.

[13] T. Scheepers, E. Kanoulas, and E. Gavves, "Improving word embedding compositionality using lexicographic definitions," in *Proc. of WWW*, 2018.

[14] T. Bosc and P. Vincent, "Auto-encoding dictionary definitions into consistent word embeddings," in *Proc. of EMNLP*, 2018.

[15] D. Aldous and J. Fill, "Reversible markov chains and random walks on graphs," 2002.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. of SIGKDD*, 2016.

[17] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.

[18] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[19] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. of EMNLP*, 2014.

[20] G. A. Miller and W. G. Charles, "Contextual correlates of semantic similarity," *Language and cognitive processes*, 1991.

[21] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *CACM*, 1965.

[22] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," *TOIS*, 2002.

[23] E. Bruni, N.-K. Tran, and M. Baroni, "Multimodal distributional semantics," *JAIR*, 2014.

[24] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch, "A word at a time: computing word relatedness using temporal semantic analysis," in *Proc. of WWW*, 2011.

[25] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren, "Large-scale learning of word relatedness with constraints," in *Proc. of SIGKDD*, 2012.

[26] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *Proc. of CoNLL*, 2013.

[27] D. Gerz, I. Vulić, F. Hill, R. Reichart, and A. Korhonen, "Simverb-3500: A large-scale evaluation set of verb similarity," *arXiv preprint arXiv:1608.00869*, 2016.

[28] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, 2015.

[29] D. Yang and D. M. Powers, *Verb similarity on the taxonomy of WordNet*. Masaryk University, 2006.

[30] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Embeddings for word sense disambiguation: An evaluation study," in *Proc. of ACL*, 2016.

[31] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*, 2007.

[32] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, 2003.

[33] S. K. Jauhar, C. Dyer, and E. Hovy, "Ontologically grounded multi-sense representation learning for semantic vector space models," in *Proc. of ACL*, 2015.

[34] D. Bollegala, T. Maehara, Y. Yoshida, and K.-i. Kawarabayashi, "Learning word representations from relational graphs," in *Proc. of AAAI*, 2015.

[35] D. Kiela, F. Hill, and S. Clark, "Specializing word embeddings for similarity or relatedness," in *Proc. of EMNLP*, 2015.

[36] I. Vulić and N. Mrkšić, "Specialising word vectors for lexical entailment," *arXiv preprint arXiv:1710.06371*, 2017.

[37] B. Gallagher and T. Eliassi-Rad, "Leveraging label-independent features for classification in sparsely labeled networks: An empirical study," in *International Workshop on Social Network Mining and Analysis*, 2008.

[38] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: graph mining using recursive structural features," in *Proc. of SIGKDD*, 2011.

[39] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. of SIGKDD*, 2014.

[40] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. of WWW*, 2015.