An Unsupervised Graph Based Continuous Word Representation Method for Biomedical Text Mining

Zhenchao Jiang, Lishuang Li and Degen Huang

Abstract—In biomedical text mining tasks, distributed word representation has succeeded in capturing semantic regularities, but most of them are shallow-window based models, which are not sufficient for expressing the meaning of words. To represent words using deeper information, we make explicit the semantic regularity to emerge in word relations, including dependency relations and context relations, and propose a novel architecture for computing continuous vector representation by leveraging those relations.The performance of our model is measured on word analogy task and Protein-Protein Interaction Extraction (PPIE) task. Experimental results show that our method performs overall better than other word representation models on word analogy task and have many advantages on biomedical text mining.

Index Terms—Natural Language Processing, Machine learning, Connectionism and neural nets, Object representation

1 INTRODUCTION

TO solve Biomedical Natural Language Processing (BioNLP) problems, words in the text should be represented into real values or real-valued vectors so that the power of mathematics can be used. The most commonly used representation method for categorical features is the one-hot coding, by which each word is represented as a vector with only one 1 and many 0s, e.g., suppose we have a dataset having only a single categorical feature "type", with values "hormone", "kinase" and "receptor", the corresponding one-hot vectors are [1, 0, 0], [0, 1, 0] and [0, 0, 1] respectively.

Experimental results show that the one-hot coding works well with learning algorithms such as Logistic Regression and Support Vector Machine. However, the pairwise Euclidean distances between them are all equal to $\sqrt{2}$, which lacks the capacity to capture the semantic regularities of words. Worse still, one-hot coding is a disaster for Euclidean distance based algorithms such as K-means. Therefore, more powerful distributed representation models are motivated.

Word representations method is an important step for most machine learning based BioNLP tasks, such as Name Entity Recognition, Protein-Protein Interaction Extraction, Drug-Drug Interaction Extraction, Event Extraction, Ontology Curation. It has been proved that reasonable distributed word representations can help improving the performance of those BioNLP tasks. Common word representation methods are summarized as follows.

Salton et al. [1] proposed VSM, an algebraic model for representing words, phrases and in general any objects as vectors of identifiers, and successfully used it in the SMART information retrieval system, which represents documents as a vector of their most important terms by means of TFIDF weighting. VSM treats a document as a bag of words (BOW) and ignores the dependence between terms, and for large corpora, high-dimensional vectors may cost a lot memory.

LSA, a matrix factorization method for generating low-dimensional word representations, utilizes low-rank approximations to decompose large matrices [2]. In LSA, the matrices are also of “term-document” type, i.e., the rows correspond to words or terms, and the columns correspond to different documents in the corpus.

- *Z. Jiang is with the School of Computer Science and Technology Dalian University of Technology, Dalian,Liaoning, China. E-mail: kukumayas@gmail.com.*
- *L. Li is with the School of Computer Science and Technology Dalian University of Technology, Dalian,Liaoning, China. E-mail: lilishuang314@163.com.*
- *D. Huang is with the School of Computer Science and Technology Dalian University of Technology, Dalian,Liaoning, China. E-mail: huangdg@dlut.edu.cn.*

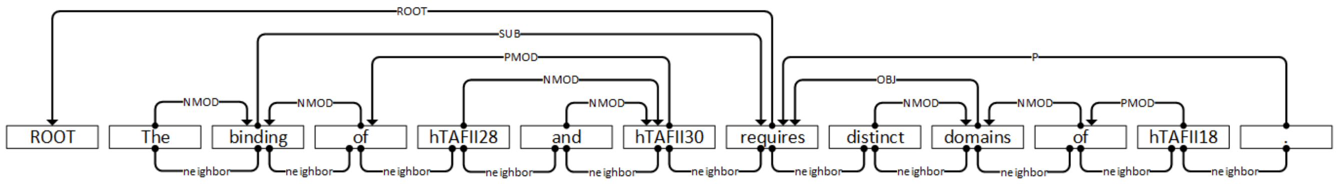


Fig. 1 An example of relation graph. The upper part is dependency relations, and the lower part is context relations.

VSM and LSA are simple models for computing a continuous degree of similarity between terms, where a document is represented as an unordered collection of words. However, a main problem with VSM and LSA is that they only consider the statistical information between words (terms) and documents, where richer information such as context words, dependency links are not considered. Besides, on large training corpus, the size of VSM and LSA vectors may be too large for the memory to load, and in this case, dimension reduction strategies should be applied. Although several techniques have been proposed to reduce the size of vectors, e.g. using Singular Value Decomposition (SVD) based technics, or simply by applying threshold based on term frequency, and richer information, such as paradigmatic information, has been included to overcome simple approaches only based on syntagmatic relationships, VSM and LSA are limited by their models to integrate richer information, such as dependency relations. Furthermore, comparing with neural network-based word representation models, VSM and LSA do relatively poorly on the word analogy task [3], i.e., they are not good enough for measuring the meaning of words. Due to these reasons, in this work, we try to improve the word representation method based on neural network leveraging richer information, to make the word vectors more powerful to express the semantics.

The idea of distributed representation was first proposed by Hinton [4], and developed by Bengio [5], Collobert [6], Mnih [7], Socher [8], Huang [9], Pennington [3], Mikolov[10], [11], [12], Levy [13], etc., which have been successfully integrated in many Natural Language Processing (NLP) tasks [3], [6] such as Name Entity Recognition (NER), Semantic Role Labeling (SRL), etc.

Socher used a semisupervised Recursive Auto Encoder (RAE) to jointly learn phrase representations, phrase structure and sentiment distributions [8] which achieved notable success to predict sentiment distributions. However, RAE was designed for sentiment prediction task, which cannot be immediately adapted to BioNLP tasks such as PPIE. Concretely, the input of RAE for the sentiment distribution prediction is a piece of text, while the input of PPIE classification model is a tuple, (protein1, protein2, sentence), which cannot be input to RAE.

Collobert and Weston's deep neural network architecture could jointly train the specified NLP task model and word representation [6], which contained a lookup table layer, a convolution layer, a max over time layer and a softmax layer. Their architecture achieved state-of-the-art performance on many NLP tasks, including Part-of-Speech (POS), Chunking (CHUNK), NER and SRL. However, their word em-beddings have been trained for about 2 months over Wikipedia, which is time-consuming.

Mikolov et al. proposed two relatively efficient architectures, Skip-Gram and Continuous Bag-of-Words (CBOW) [10], [11], [12], based a shallow window of context words, i.e., they scanned a context window across the entire corpus, and trained target word by words within the fix-sized context window. Different from joint learning, Skip-Gram and CBOW trained word representations independently in an efficient way. However, these models considered only the context words and failed to take advantage of the vast amount of different kinds of relations among words.

Pennington et al. [3] leveraged statistical information by training only on the nonzero elements in a word-word co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus to train word representations, and proposed the GloVe model which combined the advantages of the two major model families in the literature: global matrix factorization and local context window methods.

Levy and Goldberg generalized Mikolov's skip-gram model with negative sampling to include arbitrary contexts, and performed experiments with dependency-based contexts [13]. However, their method focused on the the generalization of contexts, while there was no corresponding improvement on the learning architecture.

RAE, C&W, Skip-Gram, CBOW and GloVe all trained the target word using a sequence of words (a sequence of words in the fix-sized context window or sequence of words in the whole sentence). As shown in Fig. 1, we note that text is not only an arrangement of words, but also a structure of dependency, which should be considered for training word representations. Although Levy.'s method generalized the contexts with dependency to train word vectors and ahieved state-of-the-art performance in their evaluation, it did not provide a corresponding improved learning architecture. In this work we analyze the relations between words according to dependency tree and neighborhood, and propose a novel three-top-layer architecture to train word representation based on relation graph, which is fast to train and have many advantages. Next we show how the new neural network-based word representation model overcoming three shortages stated above: 1) it leverages dependency and context relations rather than only considering context window, and 2) it is task irrelevant rather than joint training, which can be integrated into any NLP task and 3) it is trained with high efficiency, for example, the word representations for PPIE can be trained within 15 minutes.

2 Methods

A word would not have any practical meaning but "an

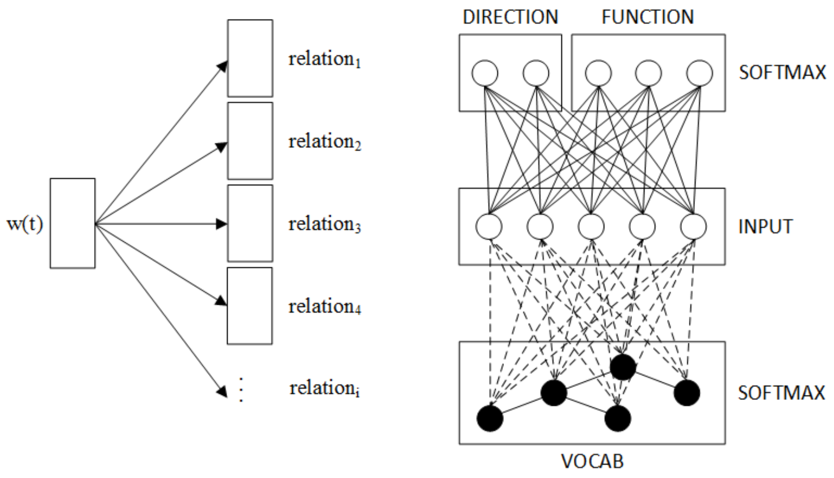


Fig. 2. The two-layer Neural Network Graph Model and the detailed learning architecture. three-top-layer.

isolated word" if it had no relations with any other words. We think the essence of semantic meaning lies in the relations, and the relations are the key for computers to understand the meaning of words. For example, the relation between "binding" and "requires" as shown in Fig. 1, means not only that there is a connection between "binding" and "requires", but also that "binding" and "requires" confirm to subject-predicate structure. Without the relation type "SUB" and "OBJ", the algorithms cannot distinguish "binding", "domains" when training "requires". Thus to capture the semantic meaning of a word, we focus on how to explain what and how other words are linked with it.

In our view, the semantic information of a word can be formalized as what and how other words are related with it. Previous methods such as NNLM and CBOW trained a word by context words, whereas in this paper, we train each target word by relations according to graph.

2.1 Neural Network Graph Model

Fig. 2 shows the neural network-based architecture. First we establish some notation. Let X be the matrix of word representation, which is randomly initialized and X_t is the corresponding vector of the tth word. Let all links in the corpus be denoted by L, Let $P_{j,t}=P(y=L_j\,|\,X_t)$ be the probability that the jth link appears in the context of the tth word. and finally let the jth link $L_j$from a target word t to the wth be denoted by $L_{t,w}$ which can be formalized as a tuple (vocab$_w$, function$_{t,w}$, direction$_{t,w}$), or (V_w, $F_{t,w}$, $D_{t,w}$) for short. For example, the word "binding" can be denoted as Table 1 shows (tuples for NNGM) according to Fig. 1.We begin with a simple example that showcases how certain aspects of meaning can be extracted directly from links. Consider a biomedical termV_t, suppose we are interested in whether it is a protein or not, for which we might take V_t = cofilin. The meaning of cofilin can be recognized by studying the probabilities with its relations. For link L_j related to cofilin, L_j= (cofilin, phosphorylate, OBJ, $\rightarrow$), we expect the conditional probability$P(y=L_j\,|\,X_t)$ will be large, because phosphorylation of proteins is an important regulatory mechanism, and large conditional probability of L_j given X_t proves that V_t is a protein that can be phosphorylated.

$$\begin{aligned} \mathrm{P}_{j,t} &= \mathrm{P}^{V}_{j,t} + \mathrm{P}^{F}_{j,t} + \mathrm{P}^{D}_{j,t} \\ &= \frac{e^{X_t\lambda^V_{L_j}}}{\sum_{k\in V} e^{X_t\lambda^V_k}} + \frac{e^{X_t\lambda^F_{L_j}}}{\sum_{k\in F} e^{X_t\lambda^F_k}} + \frac{e^{X_t\lambda^D_{L_j}}}{\sum_{k\in D} e^{X_t\lambda^D_k}} \\ &= \sum\nolimits_{C\in\{V,F,D\}} \frac{e^{X_t\lambda^C_{L_j}}}{\sum_{k\in C} e^{X_t\lambda^C_k}} \end{aligned} \qquad (1)$$

The above argument suggests that the appropriate starting point for word vector learning should be with the conditional probabilities. Links implicate the meaning of words byV_w (the word V_w that$V_t$links to), $F_{t,w}$ (the grammatical function of the link) and $D_{t,w}$ (the direction of the link). Given a target word vector X_t, we want our hypothesis to estimate the probability $P_{j,t}$ for each link by the three parts. Thus, our hypothesis will output a $|V|+|F|+|D|$ dimensional vector whose elements sum to 3. Concretely, the conditional distribution takes the form

where λ^V, λ^F, λ^D are the weight matrices of softmax layer for vocab, function and direction (refer to Fig. 2) respectively, and X^T is the transpose of the input matrix X. Parameters λ^V, λ^F, λ^D and word vector X_t are trained to minimize the cost function

$$J = -\sum\nolimits_{C\in\{V,F,D\}} \sum\nolimits_{k\in C} 1\{y_C = C_k\} \log \mathrm{P}^C_{k,t} \qquad (2)$$

where 1{·} is the indicator function, 1{a true statement}=1, and 1{a false statement}=0.To solve for the minimum of J and train the word vectors X_t, we resort to gradient descent, the widely used iterative optimization algorithm

$$\lambda^C_{L_k} \leftarrow \lambda^C_{L_k} - \alpha\nabla_{\lambda^C_{L_k}} J \qquad (3)$$

$$X_t \leftarrow X_t - \alpha\nabla_{X_t} J \qquad (4)$$

Where α is the learning rate. Taking derivatives, the gradients are

$$\nabla_{\lambda^C_{L_k}} J = -X_t(1\{y_C = C_k\} - \mathrm{P}^C_{k,t}) \qquad (5)$$

$$\nabla_{X_t} J = -\sum\nolimits_{C\in\{V,F,D\}} \sum\nolimits_{k\in C} 1\{y_C = C_k\} \left(\frac{\sum_{i\in C}\left(\lambda^C_{L_k} - \lambda^D_i\right) e^{X_t\lambda^C_i}}{\sum_{i\in C} e^{X_t\lambda^C_i}} \right) \qquad (6)$$

2.2 Hierarchical Softmax

Although two-layer architecture seems to be shallow, to deal with millions of sentences, it is not efficient enough because of the tremendous computing and updating workload of forward-propagation and back-propagation. Morin and Bengio introduced a hierarchical decomposition of the conditional probabilities that yielded a speed-up of about 200 built by using WordNet [5]. Mikolov used a similar strategy where the vocabulary was represented as a Huffman binary tree, based on the observations that the frequency of words worked well for obtaining classes in neural net language models [14].

WordNet does not contain complete vocabulary of training corpus, therefore we choose Mikolov's version. With binary tree representations of the vocabulary, the number of prediction units that need to be evaluated can go down to around $log_2|V|$.

To adopt hierarchical softmax, we first build a Huffman binary tree according to the frequency of words. Concretely, sort the word list by frequency and make the twolowest elements into leaves, creating a parent node with a frequency that is the sum of the two words' frequencies, repeat this step until all words are included in the tree. All words are leaf nodes, and words with low frequency have high depths and long binary codes.

Actually, hierarchical softmax is a special multiclass logistic regression. Concretely, Let *N* be the nodes on the path from the root to target word *t*, during the training of X_t, only a submatrix of weight matrix λ is needed for forward-propagation and back-propagation, denoted by λ_N. The probability takes the form

$$\mathrm{P}_i(X_t) = \frac{1}{1+e^{-X_t\lambda_i}}, \quad i \in N \tag{7}$$

In the same way as softmax regression, the cost function of hierarchical softmax

$$J = -\sum_{i\in N}(y_i \log P_i(X_t) + (1-y_i)\log(1-P_i(X_t))) \tag{8}$$

is minimized by means of gradient descent:

$$\lambda_i \leftarrow \lambda_i - \alpha \sum_{i\in N} X_t(y_i - P_i(X_t)) \tag{9}$$

$$X_t \leftarrow X_t - \alpha \sum_{i\in N} \lambda_i(y_i - P_i(X_t)) \tag{10}$$

3 Experiment

The objective of this paper is to train word vectors that represent more semantic regularities and improve the performance on biomedical text mining. Although those word vectors can be integrated into any NLP application and might further improve its performance, we cannot evaluate them on each NLP task in this paper. Therefore, to evaluate the performance of NNGM and compare it with other word representation models, first, we use an evaluation scheme proposed by Mikolv et al. [12], i.e., the word analogy task, to examine how much semantic regularity are learned, and second, we integrate the word representations into PPIE, to examine how much the extraction of PPIs can be improved by using word vectors. In this section, we describe the details of evaluation.

3.1 Evaluation Methods

Word analogy. The word analogy task consists of questions like, "*a* is to *b* as *c* is to __?". The dataset contains 19544 such questions, divided into 14 categories, and each category contains a semantic subset and a syntactic subset. The semantic questions are typically analogies about people or places, like "Athens is to Greece as Berlin is to __?". The syntactic questions are typically analogies about verb tenses or forms of adjectives, for example "dance is to dancing as fly is to __?". To correctly answer the question, the model should uniquely identify the missing term, with only an exact correspondence counted as a correct match. We answer the question "*a* is to *b* as *c* is to __?" by finding the word *d* whose representation v_d is closest to $v_b - v_a + v_c$ according to the cosine similarity.

PPIE. PPIE aims to find a criteria to judge whether a pair of proteins actually implies interaction or not according to the biomedical text that mentions them. Five publicly available PPI corpora extracted from MedLine contain interaction annotation: AIMed [15], BioInfer [16], HPRD50 [17], IEPA [18], LLL [19]. For example, according to a description from AIMed, "The binding of hTAFII28 and hTAFII30 requires distinct domains of hTAFII18", one can infer that hTAFII28 interacts with hTAFII30. To automatically extract protein interactions, the model should classify the PPI candidates into two groups, positive ones and negative ones. In order to evaluate the performance of word representation, reducing the influence by crafted features and tricks like kernel methods, we extract shallowword features and use a L1 regularized logistic regression (L1-LR) based binary classification model to address the problem.

All evaluation results will be reported using the F-score. For PPIE, we perform pair-wise 10-folds cross-validation (randomly partitioned) on each corpus and report the macro-average F-score. Precision (P) is the ratio between the number of PPIs correctly detected and the total number of PPIs that were found by the system. Recall (R) is the ratio between the number of PPIs correctly detected and the total number of PPIs in the gold standard. F-score is the harmonic mean of precision and recall.

$$\mathrm{F} = \frac{2\times \mathrm{P}\times \mathrm{R}}{\mathrm{P}+\mathrm{R}} \tag{11}$$

3.2 Corpora and Training Details

We train word representations on 20 corpora of varying sizes: 19 corpora with sentences ranging from 10 thousands 1,000 thousands, extracted from MedLine on the theme of protein; and the *text8* corpus, which is the first 10^8 bytes of the English Wikipedia dump on Mar. 3, 2006. We tokenize each corpus by making every special character ([ ] ( ) { } ' " . , + - _ * \ | / ; : ! ? = >< ~ ` # $ % ^ &) be a single token, and then we parse the text by GDEP [20] to construct the graphs for the training of NNGM.

For the training of word representation models, including Skip-gram, CBOW, GloVe, Levy's method and

Table 1. Numbers of positive instances of five PPI corpora, AIMed (A), BioInfer (B), HPRD50 (H), IEPA (I) and LLL (L).

	A	B	H	I	L
#Pos	1000	2531	163	335	164
#Neg	4834	7129	270	482	166
#Total	5834	9660	433	817	330

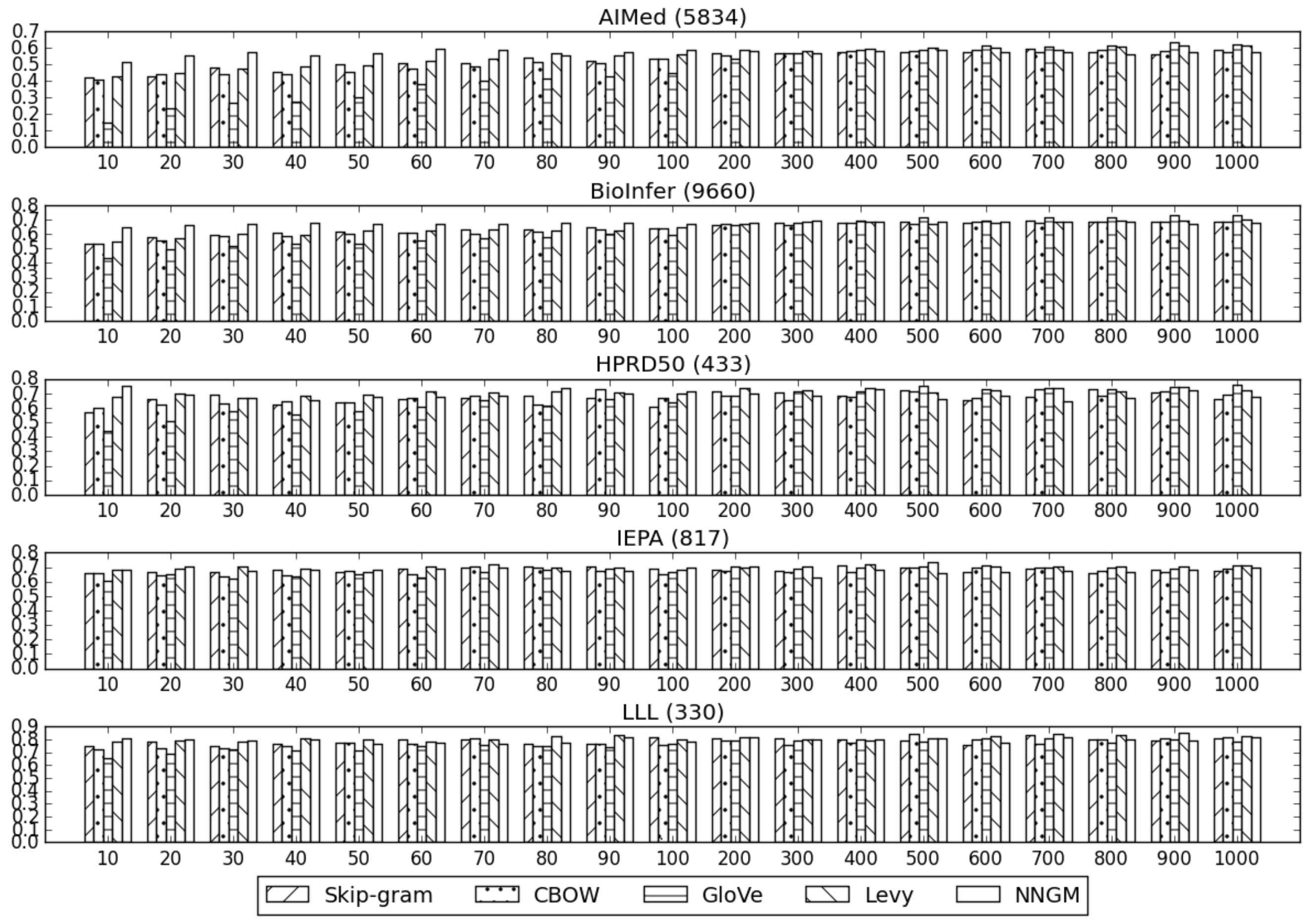


Fig. 3 A comparison of Skip-Gram, CBOW, GloVe, Levy's method and NNGM, trained on corpora from 10 thousand sentences to 1000 thousand sentences evaluated on AIMed, Bionfer, HPRD50, IEPA and LLL.

NNGM, to keep consistent with Mikolov's word2vec, we initialize X with same randomization method, and coefficient λ with zeros, initial learning rate of 0.025, context window of 5 and min-count of 0.

For the evaluation on PPIE, our goal is not to get higher F-scores than previous works by tweaking the feature set or choosing advanced classification algorithms. We concern about the performance of word vectors, and whether the classification of PPIs can be improved by leveraging word representation or not. Therefore, we only use surface word features for PPIE:

1. Entity Name. Words in the target protein pair.
2. Surrounding Words. Words that surrounds the target proteins.
3. Inner Words. Words between the target proteins .
4. Sentence. All words in the sentence.

We first obtain the word vectors by different models, and then transform the PPIE feature vectors to numerical input vectors, and finally classify them with the L1-LR based model.

Table 2. Evaluation of 5 word representation models on Word Analogy task. The evaluation corpus is from English Wikipedia dump on Mar. 3, 2006, which can be downloaded at http://mattmahoney.net/dc/textdata

	Total	Semantic	Syntactic
GloVe	2.27	2.19	2.3
SG	25.91	26.05	**25.84**
CBOW	16.13	9.41	19.51
Levy	6.22	2.76	9.01
NNGM	**26.41**	**33.17**	23.32

4 Result

In this section, we present the results on word analogy and PPIE, comparing with Skip-Gram, CBOW, GloVe, Levy's method and NNGM.

4.1 Word Analogy Evaluation

To get better performance on word analogy task, previous works trained different word vectors on different corpora. For example, Pennington et al. [3] trained GloVe on five corpora varying sizes: a 2010 Wikipedia dump with 1 billion tokens; a 2014 Wikipedia dump with 1.6 billion tokens; Gigaword 5 which has 4.3 billion tokens; the combination Gigaword5 + Wikipedia2014, which has 6 billion tokens; and on 42 billion tokens of web data, from Common Crawl5, while Mikolov [12] and Levy [13] used other training corpus. Larger training corpus provides better statistics, however, we think it is not fair for the comparisons with the word representation models, and it is better to evaluate different models by training word vectors using same corpus. Therefore, we use the test data for the Large Text Compression Benchmark for evaluation, which was also used by Mikolov et al [12].

As Table 2 shows that, NNGM model performs the best (26.41), especially on semantic task (33.17), which proves that our three-top-layer architecture which levarages dependency relations and context relations is good for learning semantic regularites.

On syntactic task, NNGM is 2.52 lower than Skip-gram model, and higher than other models. Note that the syntactic questions are typically analogies about verb tenses or forms of adjectives, for example "dance is to dancing as fly is to __?", which has nothing to do with syntactic parsing or syntactic information. The incorporation of dependency relations of NNGM helps to learn the role of words, while weakens their form, i.e., for NNGM, the training of a word considers less about whether it is singular or plural, while considers more about its dependency role. Therefore, it is understandable that NNGM performs better on semantic task than on syntactic task.

GloVe model does poor in our experiment, which may due to that the training on *text8* does not make full use of GloVe model, since one of the advantage of GloVe model is leverageing statistical information. GloVe could be more advantagous on larger corpus, e.g., Gigaword5 + Wikipedia2014 with 6 billion tokens, which provides better statistics while taking much more training time.

Levy's method also leverages dependency information, however the result also poor, which we think mainly due

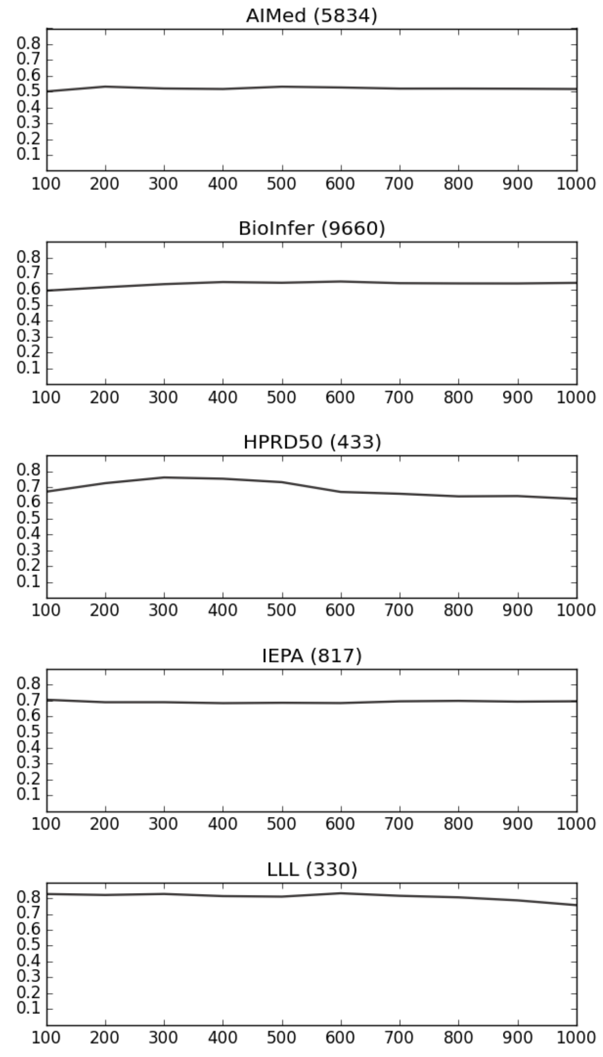


Fig. 4 Comparison of different vector dimensions, trained using NNGM, evaluated on AIMed, Bionfer, HPRD50, IEPA and LLL.

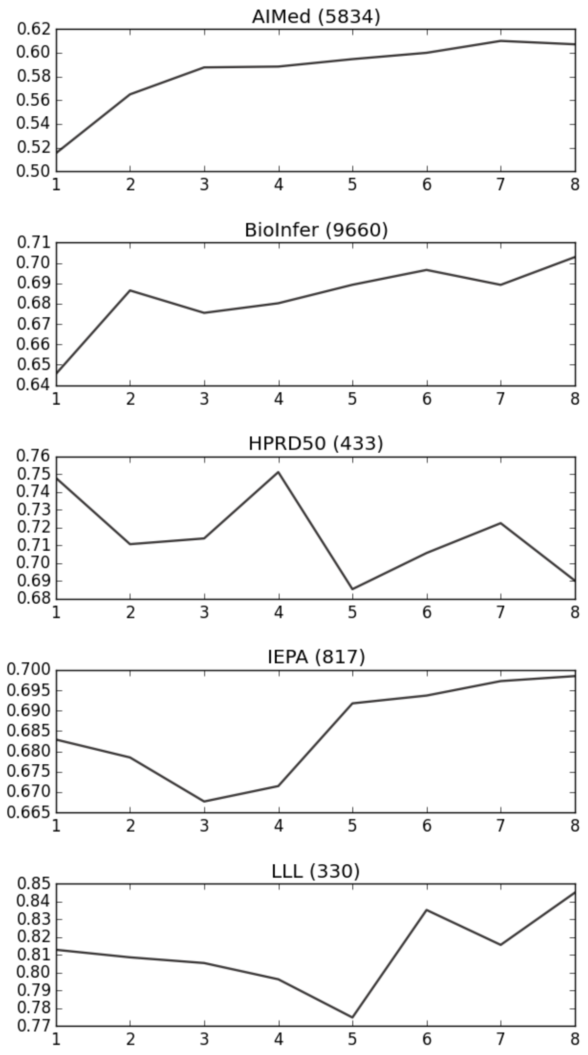


Fig. 5 Comparison of training iterations, trained using NNGM, evaluated on AIMed, Bionfer, HPRD50, IEPA and LLL.

to two reasons: First, Levy's method increases the sparsity of context, for example, suppose that there are *M* words in vocabulary, *N* types of relation, *O* directions, there will be *M*N*O* nodes in the hierarcchical softmax tree for Levy's method, which weakens the power of hierarchical softmax, while for NNGM, the number of nodes is only *M+N+O*. Second, Levy's method used Mikolov's two-layer architecture without any corresponding improvement for dependency which seems not very reasonable. For example, suppose to train the word "australian" using context relations (scientist, AMOD, ->) and (scientist, AMOD, <-), for Levy's method, they correspond to totally different nodes in the binary tree, ignoring the fact that the word (scientist) and relation type (AMOD) are the same, while NNGM overcomes this shortage by using three corresponding top layers.

4.2 PPIE Evaluation: Corpus Size

In Fig. 3, we show the performance on PPIE task for word vectors trained on the 19 corpora using GloVe, Skip-gram, CBOW, Levy's method and NNGM respectively.

First, we observe that for GloVe, Skip-gram, CBOW and Levy's method, they all have clear ascending periods on AIMed and BioInfer, from 10 thousands to 100 thousands, while NNGM has as good performance as on large corpora constantly. The experimental results on AIMed and BioInfer from 10 thousands to 100 thousands sentences show that NNGM has absolute advantages when using small corpus. This is very useful for applications with scarce resource, especially for non-English languages, for example, electronic medical record, traditional Chinese medicine side effect dataset, English-Oromo bilingual corpus, and a certain Kaggle competition, the data of which includes a text field of description of projects, because one does not need to collect too much background text to train word vectors since NNGM can perform well when using small training corpus.

Second, all the word representation models except GloVe all have clear upper bounds. We can see from Fig.3 that on AIMed, BioInfer, HPRD50, IEPA and LLL, the F-scores slightly change after 100 thousands. Therefore, it is very important for a word representation model to reach the upper bound as quick as possible.

Third, for large training corpora, NNGM performs generally comparable with other word representation models, while on small corpora, NNGM is overall better.

4.3 PPIE Evaluation: Vector Length

Both Mikolov et al. and Pennington et al. claimed that higher dimensional word vectors will improve the accuracy [3], [12]. In another word, for Skip-gram, CBOW and GloVe, low dimensional word vectors cannot fully learn the semantic regularities. For example, the dimensions larger than 300 are significantly better than 100 for GloVe model, and Skip-gram and CBOW also perform well on PPIE when dimension larger than 400. Therefore, all experiments in this paper on PPIE use 400 as the vector dimension for all Skip-gram, CBOW, GloVe, Levy's method and NNGM to make sure that the experimental results are fair.

Actually, unlike the other four models, NNGM does good even using small vector dimension. As shown in Fig. 4, on AIMed, BioInfer, HPRD50, IEPA and LLL, the F-scores are hardly influenced by the dimension of word vectors trained using NNGM. The performance of at 100 dimension is almost the same as 1,000 dimension (AIMed, BioInfer, IEPA), and sometimes even better (HRPD50, LLL).

Based on these findings, we conclude that, first, unlike the other four models, NNGM learns semantic regularites well even when word vector dimension is small, which can save training time. Second, using NNGM, one does not need to try so many dimensions to find an optimized one. This is very important for practical NLP applications, for example, when we integrate Skip-gram into PPIE, we tried vector dimensions from 100 to 1,000 and conducted a large number of experiments to find the best one, which took a lot of time, while for NNGM, we can simply use 100. Third, low dimensional word vectors make the NLP program occupy less memory, this is important especially for learning algorithms such as Deep Learning since these algorithms often need a lot of memory.

4.4 PPIE Evaluation: Iteration

Besides corpus size and vector length, another important factor that influences the word vector is iteration. Increasing training iterations can also improve the performance of word vectors. This is very useful for applications with scarce resource like electronic medical record, traditional Chinese medicine side effect dataset, and for improving the performance of NLP applications.

As shown in Fig. 5, generally speaking, the performance of word vector is better when the number of iteration increases. For example, trained using 10 thousands sentences and evaluated on AIMed, the F-score of NNGM is 60.6 when iteration is 8, which is 9.1 higher than when iteration is 1, and also higher than Skip-gram (59.0) and CBOW (57.6) trained using 1,000 thousands sentences with 1 iteration. Based on these observations, we recommend increasing interations to train better word representations for NLP applications such as PPIE when using NNGM.

4.5 Efficiency

The total run-time is split between constructing graphs and training the models. The former is decided by the size of training text and also influenced by the efficiency of GDEP. The training speed depends mostly on the size of training text, the number of epochs and the vector size, and we explore the effect of these choices in following sections in detail. The training complexity of NNGM are $VectorSize \times (3+|D|+|F|+\log_2|V|)$.

5 Discussion

5.1 PPIE Corpora Differences

Although NNGM performs well on PPIE tasks, we observe that its performance varies on different corpora. Concretely, the learning curves are ideal on AIMed and BioInfer, which contain more than 5,000 instances, and while the learning curves are sometimes unusual on HPRD50, IEPA and LLL, which contain less than 1,000 instances. The differences of corpora mainly include:

1. Corpus size. As Table 1 shows, AIMed and BioInfer are much larger than the other three corpora, therefore, we consider more about the experimental results on AIMed and BioInfer, since the other three corpora suffer more from sparsity. Generally, the results on large corpora are more convincing.
2. Annotation style. AIMed, BioInfer, HPRD50, IEPA and LLL are annotated by different annotators, which may lead to the different input distribution and performance for different corpora.
3. Label-bias. As we know that the ratio of postive instances to negative instances influences the performance of machine learning algorithms. As shown in Table 1, the ratios are variant on the five corpora.

The above factors potentially affect the distribution of input and further influence the results. Generally speaking, larger corpora suffer from less sparsity, therefore, the results on AIMed and BioInfer are more convincible.

5.2 Major Findings

In this work, we propose a two-layer neural network which has three top layers to leverage dependency relations and context relations between words to unsupervisely train word representation for BioNLP. NNGM leverages dependency relations and context relations rather than fix-sized context window of words. Given word *t*, the architecture calculates the likelihood of relations that involves *t*, considering three parts: vocabulary (V), function (F) and direction (D), and word vectors are updated during the estimation of parameters. Major findings are summarized as follows：

First, from Table 2 we can see that NNGM performs well on word analogy task, especially on semantic task. The incorporation of dependency relations by using three top layers helps the learning of the role of words. Trained on same unlabeled text, the NNGM performs overall better than Skip-Gram and CBOW as shown in Table 2, which proves that dependency relations and context relations are richer than context words for training word representations. Therefore, NNGM is recommended for word analogy related NLP applications.

Second, it is almost impossible to judge which one of the five word representation models, Skip-gram, CBOW, GloVe, Levy's method and NNGM, is the best one under all circumstances. Choosing word representaton model for specific NLP applications is like choosing machine learning algorithm for specific data. Therefore, we conduct a set of experiments to give suggestions on how to choose appropriate word representation models for your applications. According to Fig. 3, it is better to use NNGM for NLP applications when the word representaiton corpus has than 200 thousands sentences, and GloVe model is recommended if more than 500 thousands sentences are available. For other situations, all the five models are worth trying.

Third, in Fig. 4, we compare the influence of word vector dimension. Unlike previous models such as Skip-gram and GloVe, the NNGM does not need high dimensions (e.g., 400) to fully learn the semantic regularites. One can simply choose 100 as the size of word vector and save computational cost for NLP applications by using NNGM.

Fourth, generally speaking, to increase iteration times can improve the performance of NNGM, for example, 8 iterations using 10 thousands sentences can achieve higher F-scores on PPIE than 1 iterations using 1000 thousands sentences. Therefore, we recommend training NNGM using more iterations if possible.

Last, NNGM uses an independent architecture to train the distributed word representation. Unlike joint learning models, one can first train word vectors using any unlabeled text by NNGM, and then embed these vectors into any NLP tasks.

6 Conclusion

We introduce a new unsupervised neural network graph model to learn distributed word representations. Our model leverages the relations between words, including dependency relations and context relations. We demonstrate that our word representation achieves state-of-the-art performance on word analogy task and PPIE. Major contributions can be summarized as follows:

1. We present a novel unsupervised word representation training model, NNGM, which is task-irrelevant and can be embedded into any biomedical text mining tasks.
2. NNGM considers richer information, i.e., dependency relations and context relations among words, and it outperforms other representation methods on word analogy task, especially on semantic subtask.
3. The word vectors trained by NNGM can be used in many biomedical text mining applications. The evaluation results on PPIE shows that NNGM has comparable performance with other word representation models when trained on large corpora, while outperforms other models when trained on small corpus.
4. Experimental results on PPIE also suggests that NNGM is not senstive to vector dimension, which makes it a good choice for saving computational cost, and more training iterations can improve the performance of NNGM.

Acknowledgment

This work was supported by the National Natural Science Foundation of China [grant numbers 61173101, 61173100]. Corresponding author is Lishuang Li, and her email address is lilishuang314@163.com

REFERENCES

[1] G. Salton, A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[2] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Am. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[3] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation." in *Proceedings of Empiricial Methods Nat. Lang. Process*, 2014.

[4] G. E. Hinton, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, 1986, pp. 1–12.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A Neural Probabilistic Language Model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[7] A. Mnih and G. E. Hinton, "A Scalable Hierarchical Distributed Language Model," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 1081–1088.

[8] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 151–161.

[9] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving Word Representations via Global Context and Multiple Word Prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, 2012, pp. 873–882.

[10] T. Mikolov, W. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," 2012.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality arXiv: 1310 . 4546v1 [cs . CL] 16 Oct 2013," pp. 1–9.

[12] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation ofWord Representations in Vector Space," pp. 1–12.

[13] O. Levy and Y. Goldberg, "Dependency-Based Word Embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, vol. 2.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv Prepr. arXiv1301.3781*, 2013.

[15] R. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, and Y. W. Wong, "Comparative experiments on learning information extractors for proteins and their interactions," *Artif. Intell. Med.*, vol. 33, no. 2, pp. 139–155, 2005.

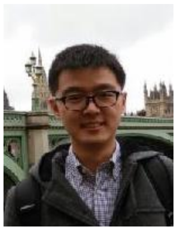
[16] S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski, "BioInfer: a corpus for information extraction in the biomedical domain," *BMC Bioinformatics*, vol. 8, no. 1, p. 50, 2007.

[17] K. Fundel, R. Küffner, and R. Zimmer, "RelEx—Relation extraction using dependency parse trees," *Bioinformatics*, vol. 23, no. 3, pp. 365–371, 2007.

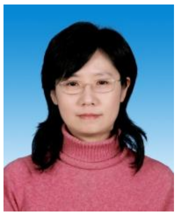
[18] J. Ding, D. Berleant, D. Nettleton, and E. Wurtele, "Mining MEDLINE: abstracts, sentences, or phrases," in *Proceedings of the pacific symposium on biocomputing*, 2002, vol. 7, pp. 326–337.

[19] C. Nédellec, "Learning language in logic-genic interaction extraction challenge," in *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, 2005, vol. 7.

[20] K. Sagae and J. ichi Tsujii, "Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles.," in *EMNLP-CoNLL*, 2007, pp. 1044–1050.



Zhenchao Jiang received his BSc degree from Shandong Normal University in 2010. He is taking a successive postgraduate and doctoral program at Dalian University of Technology. He has involved in many projects and has published many research papers in recent years. His research interests include data mining, machine learning and text analysis. Recently, he is focusing on text representation and neural network algorithm for natural language processing applications.



Lishuang Li received her BSc degree from Dalian University of Technology in 1989, the MSc degree from Dalian University of Technology in 1992, and the PhD degree from Dalian University of Technology in 2013. She is currently a professor in School of Computer Science and Technology at Dalian University of Technology. She has published more than 60 research papers in various journals and conferences in recent years. Her research interests include text mining, natural language processing and machine translation. In recent years, she has focused on text mining for biomedical literatures and information extraction from huge text resources. Her research projects are funded by the NSFC.

Degen Huang received his MSc degree from Dalian University of Technology in 1988 and his PhD degree from Dalian University of Technology in 2004. He is currently a professor in School of Computer Science and Technology at Dalian University of Technology. His research interests include machine translation, text mining, natural language processing and machine learning. In recent years, His research projects are funded by the NSFC.