Short Text Classification Based on Graph Representation and Deep Neural Network

Zibin He
School of Data and Computer Science
Sun Yat-sen University, China

Abstract

The performances of conventional text classification methods are supported by large-scale corpus with rich semantic information, which is what short texts basically neglect. To remedy this, we introduce two levels of semantics enriching: word level and text level. In both of the levels, distributed representations can be generated via network embedding algorithms. Word vectors and text vectors are obtained from the graph built from, respectively word co-occurrence matrix and text similarity matrix. Representations from different levels jointly train a convolution neural network. Empirical results show our proposed method outperforms the state-of-the-art text classification methods.

Introduction

Text classification plays a crucial role in many natural language processing applications, such as web search, user-based recommendation. The core problem of text classification tasks is how to represent variable-length documents into a fixed-length vector and capture as much semantic information as possible. Recent years have witnessed an increasing number of research on the distributed representation of continuous words, such as Bengio et al NNLM[1], word2vec[3, 4], GloVe[5]. With the learned document vectors, multiple algorithms can be applied to address the classification problems, like support vector machine, logistic regression or even deep neural network[2, 6, 8, 9, 10].

However, the distributed representations call for large corpus with sufficient semantics. With respect to short texts, basically they come from social media like Twitter and their averaged length is approximately 20. In addition, they are characterized by sparsity, insufficient semantics and above all keyword orientation, leading to poor performances in existing approaches. With the proliferation of the Internet, short texts have become the carrier of information. Improving the performance of short-text classification is significant and the accuracy may determine the quality of service to some extent.

Our technique is inspired by the recent work in network embedding[11, 12, 13], which learns latent node vectors of a graph. In their theory, nodes in a graph can be regarded as words in corpus and the sequences generated from random walk is similar to a sentence. With this formulation, network embedding can be solved using any methods of learning word distributed representations.

In this paper, due to the weakness of short texts, we propose two levels of semantics enriching. In word level, we choose a specific context window size to generate the word co-occurrence matrix and the corresponding graph subsequently. We also modify the random walk strategy to exclude words with a low variance. In text level, we build a graph with nodes representing documents and edges signifying the number of common words between them. Network embedding algorithms can be employed to learn word-level and text-level representation respectively. Moreover, another set of word vector can be learned via the original corpus. Instead of directly concatenating two sets of word-level vector, we seek for a better strategy to combine the learned features. We feed them into a convolution neural network to learn a new combined vector. Finally, we concatenate the text-level vectors and the new vectors and send them as input to a fully connected neural network.

Empirical results show our learned representations from word-level and text-level semantics enriching have a slightly better accuracy compared to other word or document embedding techniques. Besides, our proposed model outperforms other text classification algorithms

The rest of the paper is arranged as followed. In Section 2, we briefly introduce related work in word distributed representation, document representation and network embedding. In Section 3, we present the details of our approach for semantics enriching. The whole architecture is shown in Section 4. We empirically evaluate the results of our experiment and compare them with other baselines in Section 5 and conclude our discussion in Section 6.

Related Work

Word Distributed Representation. Distributed representations of words have been widely used in many natural language processing tasks and help achieve better performance of learning algorithms. Conventional methods, like Bag-Of-Word, treat words as atomic tokens, which takes no consideration of the notion of similarity and order between words, as words are represented as indices in a vocabulary. Even though n-grams method takes into account the local order of words, the dimension of a word is related to the size of vocabulary, resulting in incredible computational overhead. Distributed representation is a low-dimensional vector of real number[1], which can capture semantic and syntactic similarities between words. For example[4], *vector("King") – vector("Man") + vector("Woman")* is a vector closest to the word vector of *Queen*. The research of continuous vector of words can be dated back to last century. Neural network language model uses a three-layer neural network to learn a statistical model, with word vectors as by-product[1]. This work was followed by other researchers and later it was shown that many NLP tasks can be addressed by applying the same architecture for training word vectors instead of specifying hand-crafted features[2]. Skip-gram model learns word representations by predicting the neighbors of a word given a specific size of window[3]. GloVe leverages the global statistical information and trains on the word co-occurrence matrix[5]. All these methods must be supported by large-scale corpus. With respect to short texts, due to insufficient semantics, the algorithms mentioned above cannot perform well. It is incredibly significant to enrich the semantics of original texts.

Document Representation. The conventional method is to average all the word vectors occurring in a document. However, this act would lose ordering of words and ignore semantics. Advanced approaches were proposed to better integrate the features of different words in a document into a fixed-length vector. TextCNN stacks word vectors into a matrix and feed it into a simple convolution neural network to extract important features[6]. Paragraph Vector is similar to Skip-gram model and trains distributed document representations by regarding text as one kind of context and predicting words in the document[7]. A Recurrent Neural Network is capable of processing a variable-length document by feeding one word at a time. One simple RNN strategy is to map the input document into a fixed-length vector as output[14]. TextRNN model can train a network jointly on multiple tasks simultaneously[8, 9].

Network Embedding. Many network analysis tasks involve predictions on nodes or edges. In a typical prediction problem over machine learning algorithms, features of nodes have to be crafted manually based on professional knowledge. Such features vary from tasks to tasks. An alternative strategy is to learn latent representation of nodes via unsupervised algorithms. DeepWalk algorithm uses local information obtained from random walks to learn node representations by predicting the neighbors given a target vertex[11]. In DeepWalk theory, the algorithm tries to maximize the conditional probability of a node occurring in a random-walk sequence of target nodes. LINE model is able to not only preserve the first-order proximity and second-order proximity while learning node

vectors, but also scale up to very large networks of any types, directed or weighted[12]. Node2vec model is similar to DeepWalk but modify the standard random walk procedure and introduce a biased second order random walk with two parameters: return parameter and In-out parameter[13]. The proposed random walk procedure can better explore various kinds of communities. Another research not only takes into account of the microscopic structure of a network, but also models the community in the learning of representation[15]. Our model modifies the random walk procedure of node2vec by adding early-stop strategy.

Semantics Enriching

Documents that conventional text classification algorithms work well in are always descriptive and semantically rich. Instead, short texts are semantically sparse and keyword-oriented. The example below is cited from Wikipedia and Twitter'comments, and indicates the difference mentioned above.

- Paris is the capital and most populous city of France, with an area of 105 square kilometers and a population of 2,206,488. Since the 17th century, Paris has been one of Europe's major centers of finance, commerce, fashion, science, music and painting.
- Paris always felt so safe to me everywhere I went. It's the most fashion city in the world.

Given the limitation of short texts, the learned word representations cannot capture too much semantic information like conventional descriptive documents do, which results in relatively poor performances in the text classification task. To address this problem, the key is to enrich the semantics of short texts. Here, we propose two semantics enriching strategies over word level and text level.

Word-level Semantics Enriching. Inspired by network embedding theory, relationships between graph and corpus can be bridged by the word co-occurrence matrix. We can find the local information

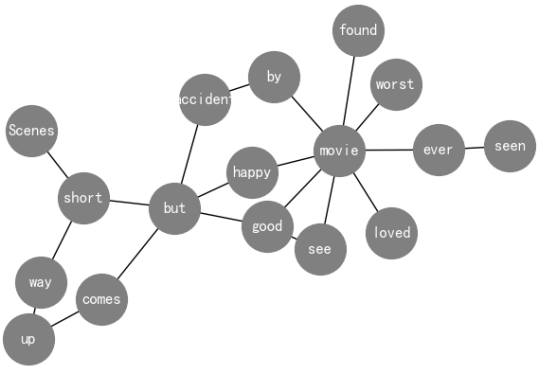

Fig. 1. Graph built from movie review

from the graph, like neighbor information and community information. Fig. 1 shows a graph built from movie review datasets, where two words occurring in a specific size of context window would connect each other.

Second order biased random walk can be executed to generate sequences of words, i.e. documents. Parameter p controls the likelihood of revisiting a node in the walk while parameter q controls the likelihood of searching for nodes outside. If p is smaller than 1, the walk would be biased toward nodes in the neighborhood of starting nodes, which can explore the underlying local relation between nodes. If q is smaller than 1, the walk is inclined to visit nodes that are far away from the starting node, which encourages outward explorations. By setting the two hyperparameters, we can control the degree of random walk's visiting outward or inward.

In Fig. 1, we observe the node of *"movie"* is a hub node, connecting multiple nodes. Since the graph is built from a movie review dataset, the word *"movie"* occurs nearly in every review. In this case, this word is not distinguishable over different labels because it connects both positive words like *"good"*, and negative words like *"worst"*. If this node is visited in the walk, the next node to be visited would be uncertain, which brings about unnecessary noises. Since these kinds of words vary in different situation, we cannot filter them out in tokenization. In addition, finding them out manually would be time-wasting and they do not generalize across different datasets.

In this paper, we propose a general random walk

Algorithm 1 GetStopwords

Input: Texts T, Variance threshold δ, Labels y

Output: *Stopwords*

initialize: Set *stopwords* to Empty
for *text* in T **do**
  for *word* in *text* **do**
    $midmap[word][y[text]] += 1$
  end for
end for
for *word* in *midmap* **do**
  Calculate variance δ_{word}
  if $\delta_{word} < \delta$ **then**
    Append *word* into *stopwords*
  end if
end for
return *stopwords*

Algorithm 2 RandomWalk

Input: Graph G, Outset s, Length l, Stopwords S

Output: *walk*

initialize: *walk* to $[s]$
for $i = 1, 2, ..., l$ **do**
  $node_{curr} = walk[i-1]$
  $node_{next} = GetNextNode(node_{curr}, G)$
  if $node_{next} \notin S$ **then**
    Append $node_{next}$ to *walk*
  end if
end for
return *walk*

procedure that can address this problem. We define words whose variance across different labels is below a threshold, as early stop words. Intuitively, a word's variance is defined as the variance of the number of time this word happens in different classes. It gives a general idea of the spread of data. A low-variance word, like *"movie"*, does not contribute to the task and we decide to stop the random walk procedure if an early stop word is visited. In other word, we only care about the words that are useful for the classification, and not about the early stop words.

In conclusion, the word-level semantics enriching can be seen as applying the new random walk strategy to node2vec algorithm. Algorithm 1, 2 specify our word-level semantics enriching.

Text-level Semantics Enriching. Existing methods for learning document representation are almost based on the words in it. They learn features by integrating word vectors in a non-linear way, such as simple neural network, convolution neural network or recurrent neural network. Since short texts are always feature-sparse and the maximum length is less than 100, word-vector-based methods are incapable of generating high-quality document representations.

In this paper, document representations are trained in an inter-document way by exploiting the similarity between documents.

First, we build a bipartite shown in Fig. 2, where circles are texts and rectangles are words. In the bipartite, connections only exist between texts and words. The bipartite can be characterized by document-word matrix, whose columns are document and rows stand for vocabulary. This matrix counts the word occurred in a document.

Second, by counting the number of common words between two documents, we can convert the bipartite into a simple graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, where V represents documents, E is the number of common words, i.e. similarity between documents. If two documents share many common words, the probability that they belong to the same class is higher. Here, we introduce similarity matrix that characterize this graph. Similarity matrix can be obtained from document-word matrix by computing the inner product of two rows of the matrix. Note that we do not consider the similarity within one document, because they are equivalent. Algorithm 3 specifies how the graph is built, it returns the adjacent matrix of a graph.

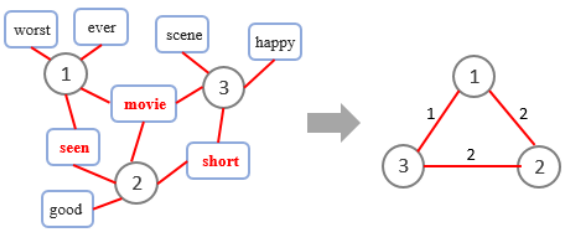

Fig. 2. Document bipartite and simplified graph

Algorithm 3 GetSimilarity

Input: Text T

Output: Similarity Matrix S

```
initialize: A to zero matrix
for text in T do
    for word in text do
        A[text][word] = 1
    end for
end for
S = A * A^T
Set diagonal to 0
return S
```

We can execute random walk procedure to generate sequences of similar documents, because random walk procedure is inclined to visit nodes that share a large weight. Network embedding algorithms can be used to learn the node(document) representations.

Conclusion. In this section, we propose two levels of semantics enriching, and consequently we obtain word representations and document representation. These supplement representations can significantly enrich the original texts and improve the performances of machine learning algorithms.

Network architecture

In the last section, we propose two level of semantics enriching, which generates one set of word vector and one set of document vector. Both of the vectors are learned from graph. Additionally, we decide to train another set of word embedding from plain texts. So far, three set of feature representations are obtained and listed as follows.

- Distributed word representations from plain texts
- Distributed word representations from graph
- Distributed document representations from graph

The document representations above are learned from similarities between documents in an inter-document way, because short texts are feature-sparse and it is difficult to extract document features directly from plain texts. However, the most commonly-used way for document representations is learned by an intra-document means, which is integrating word features into document features. Since semantics enriching has been finished, we have sufficient ingredient for building document representations from words. Here, we employ convolution neural network to learn aggregated features from two sets of word representations.

Convolution neural network is originally invented for computer vision and it shows its power in many CV tasks like object detection, face recognition. Convolution neural network utilize layers with convolving filters that can extract local features. Since filters' parameters share across different locations in an image, CNN models can easily process high dimensional data.

Subsequently, CNN models have been shown to be effective for NLP and have achieved excellent performances in many NLP tasks. The architecture for integrating two sets of word representations are discussed below. Let v_i be a n-dimensional vector for the *i*-th word in a document. The document can be represented by stacking v_i vertically. We set a maximal length of documents m and if documents are too long, we clip them to length m, and if documents are short, we pad them with zero. For every document, we can obtain a 3-dimensional tensor with dimensionality $m \times n \times 2$.

Convolving filters $\mathrm{w} \in \mathrm{R}^{\mathrm{h}\times\mathrm{n}\times 2}$ are applied to input tensors to generate new features, where h represents the size of context window. Note that two in three dimensions of filters are fixed and the only variable is h, which aims to extract the relationship between words. New features after convolving are

$$\mathrm{c} = [\mathrm{c}_1, \mathrm{c}_2, \dots, \mathrm{c}_{\mathrm{m}-\mathrm{h}+1}]$$

Then we apply global max pooling operator to select the maximum value in new feature c. We consider the maximum value as the most important feature.

As stated above, one filter can extract one value, i.e. new feature. We apply multiple filters to capture different features and concatenate them into a single vector, i.e. the new document representation.

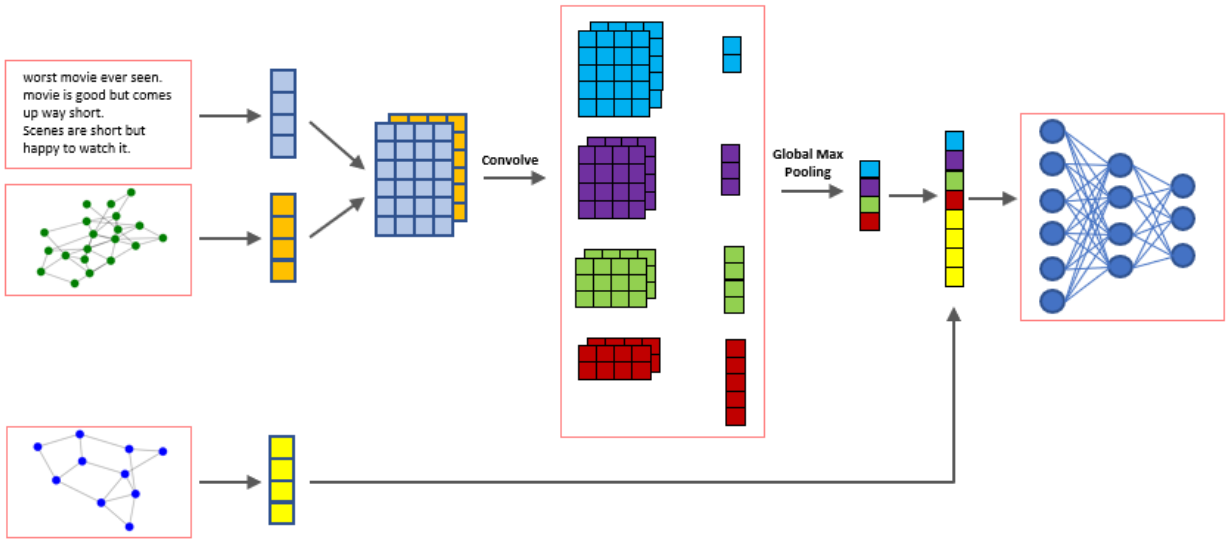

Fig. 3. The whole architecture

This new document representation is concatenated with the one from graph. Then it is fed into a fully-connected neural network with a softmax output layer. The output of the whole architecture is the probability distribution over different labels.

Conclusion. After semantics enriching, we integrate two sets of word representation to generate a new document vector with the help of convolution neural network. Two sets of document representations are concatenated together and fed into a fully connected neural network to address the short text classification problem.

Experiment

We perform experiments to better understand the effectiveness of our new proposed method. The objectives of experiments are:

- Verify the effectiveness of semantics enriching
- Verify the effectiveness of the classification method based on neural network

The experimented dataset is the AG's news topic classification dataset, which contains 120,000 samples from four classes: World, Sport, Business and Sci/Tech. Each class contains 30,000 samples. The averaged number of word of each news is approximately 20.

Word-Level Semantics Enriching. To verify the effectiveness of word representations from word co-occurrence graph, we compare them with the word vector from skip-gram model. We define V_{word} and $V_{word-node}$ as the word representations from plain text and graph respectively. These two sets of word vectors are utilized to address the classification task and the performance would be measured by accuracy. Note that the parameters of the employed classifiers are fixed and the input to classifiers is variable.

We first determine the optimal parameters of our word-level enriching method by search. The parameters include context window size and random walk length. Context window size specifies the distance between words to be similar. The larger context windows are, the more semantics and associations we can explore and the better classifiers can perform. In terms of random walk length, the further we explore, basically, the more reasonable the generated sequences are. A longer exploration length cannot always guarantee better performance due to unexpected noise. Based on the experimental results(Fig. 4), we set context window size to be 3, walk length to be 15. Other parameters of node2vec are not in our concern. Therefore, return parameter p

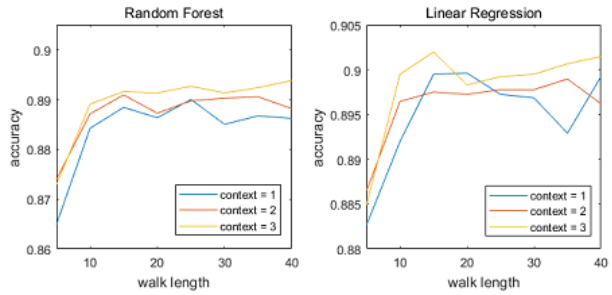

Fig. 4. Accuracy across different parameter setting

is 0.8 and in-out parameter q is 2.5, because we want closer community information to be explored.

Then we perform experiments to measure the performance of the word representations V_{word} and $V_{word-node}$ and the combination. Results in Table.1 show that $V_{word-node}$ is not effective as V_{word} probably because the generated sequences from random walk are affected by noise. However, the combination of $V_{word-node}$ and V_{word} outperforms V_{word}, which indicates the effectiveness of word-level semantics enriching. The smaller dimensions the word representation is, the better our word-level semantics enriching is. When the dimension is large enough, like 100, the increase of accuracy by performing word-level semantics enriching is trivial.

Text-level semantics enriching. The mechanism is similar to the way in verifying word-level semantics enriching. We compare the text vector with the LDA encoding (Latent Dirichlet Allocation) which is a popular generative statistical model. LDA generates

dim	input	RF	LR	XGB
10	V_{word}	0.8657	0.8680	0.8751
	$V_{word-node}$	0.8512	0.8570	0.8636
	combined	**0.8785**	**0.8900**	**0.8934**
50	V_{word}	0.8968	0.90473	0.9065
	$V_{word-node}$	0.8827	0.8925	0.8951
	combined	**0.8981**	**0.9101**	**0.9117**
100	V_{word}	0.9015	0.9101	0.9126
	$V_{word-node}$	0.8915	0.9020	0.9013
	combined	**0.9018**	**0.9114**	**0.9139**

Table. 1. Results of word-level semantics enriching

dim	input	RF	LR	XGB
10	LDA	0.7312	0.7379	0.7488
	$V_{text-node}$	0.8275	0.8613	0.8736
50	LDA	0.6902	0.7237	0.7250
	$V_{text-node}$	**0.8311**	0.8728	**0.8843**
100	LDA	0.6743	0.7139	0.7201
	$V_{text-node}$	0.8116	**0.8772**	0.8791

Table. 2. Results of text-level semantics enriching

automatic summaries of latent topics in terms of probability distribution over words for each topic, and document distributions over topics.

First, we also search for the best parameter configuration for the network embedding algorithm. It turns out that walk-length of 50 and walk-num of 20 is the best. Then we compare the text representation from node2vec and the LDA embedding code. Results in Table.2 show that our text-level semantics enriching can significantly improve the performance of classification. Among three dimensions, text vectors from dimension of 50 performs best.

Whole model. After verifying the effectiveness of our two-level of semantics enriching strategies, we evaluate the performance of our architecture by comparing with other the-state-of-art text classifiers. The configurations of our method have been determined in the previous experiments.

Note that $V_{word} + V_{word-node} + V_{text-node}$ means average the two sets of word vectors and concatenate it with text vectors. Results in Table.3 reveal that our model outperforms other methods, especially tree-based algorithms.

Model	Accuracy
FastText	0.913947
TextCNN	0.919079
RF($V_{word} + V_{word-node} + V_{text-node}$)	0.867237
LR($V_{word} + V_{word-node} + V_{text-node}$)	0.893026
XGB($V_{word} + V_{word-node} + V_{text-node}$)	0.894211
Our model	**0.922368**

Table. 3. Results of different models

Discussion and Conclusion

In this work, we proposed two-level of semantics enriching strategies given that short texts tend to be keyword-oriented and feature-sparse. The semantics enriching is implemented via network embedding algorithms. Additionally, convolution neural network is utilized for feature extraction and classification. Our model obtains better performance compared to recently-proposed methods inspired by deep learning. We will publish our code so that the community can perform research on top of our work.

References

[1] Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137 - 1155, 2003.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research, 12:24932537, 2011.

[3] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[J]. Computer Science, 2013.

[4] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[J]. Advances in Neural Information Processing Systems, 2013, 26:3111-3119.

[5] Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation[C]// Conference on Empirical Methods in Natural Language Processing. 2014:1532-1543.

[6] Kim Y. Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.

[7] Le Q, Mikolov T. Distributed representations of sentences and documents[C]// International Conference on International Conference on Machine Learning. JMLR.org, 2014:II-1188.

[8] Liu P, Qiu X, Huang X. Recurrent neural network for text classification with multi-task learning[C]// International Joint Conference on Artificial Intelligence. AAAI Press, 2016:2873-2879.

[9] Liu P, Qiu X, Chen X, et al. Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents[C]// Conference on Empirical Methods in Natural Language Processing. 2015:2326-2335.

[10] Joulin A, Grave E, Bojanowski P, et al. Bag of Tricks for Efficient Text Classification[J]. 2016:427-431.

[11] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: online learning of social representations[C]// Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. ACM, 2014:701-710.

[12] Tang J, Qu M, Wang M, et al. LINE: Large-scale Information Network Embedding[C]// International World Wide Web Conferences Steering Committee, 2015:1067-1077.

[13] Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks[C]// Acm Sigkdd International Conference on Knowledge Discovery & Data Mining. NIH Public Access, 2016:855-864.

[14] Chung J, Gulcehre C, Cho K H, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[J]. Eprint Arxiv, 2014.

[15] Wang X, Cui P, Wang J, et al. Community Preserving Network Embedding[C]// The, AAAI Conference on Artificial Intelligence. 2017.