Paper

Semantic relation classification through low-dimensional distributed representations of partial word sequences

Zhan Jin\textsuperscript{1a}), Chihiro Shibata\textsuperscript{1b}), and Kazuya Tago\textsuperscript{1c})

\textsuperscript{1} School of Computer Science, Tokyo University of Technology
1404-1 Katakuramachi, Hachioji, Tokyo 192–0914, Japan

\textsuperscript{a}) d2113002a2@edu.teu.ac.jp
\textsuperscript{b}) shibatachh@edu.teu.ac.jp
\textsuperscript{c}) ktago@edu.teu.ac.jp

Received April 9, 2018; Revised August 23, 2018; Published January 1, 2019

Abstract: By virtue of recent developments in machine learning techniques, higher-level information can now to be extracted from massive data. In this paper, we focus on extracting multiple semantic relations, using light-weight processing through the efficient low-dimensional expression of substrings in text data. We propose an approach to build features for relational classification consisted of only the low-dimensional vectors representing substrings between words called \textit{substring vectors} \cite{1}. In addition, we investigate the relationship between the numbers of dimensions and the obtained accuracies when nonlinear classifiers are applied. The experimental results show that, with simple features and small computational cost, our approach using relatively low-dimensional representations achieves a sufficiently high accuracy that is better than most existing approaches.

Key Words: neural network, language model, word vectors, nonlinear classifier, dimension reduction

1. Introduction

The popularity of the internet and computers, data now become massive and public in our society. In recent years, beyond the notion of big data, representation of data that is more efficient for intelligent systems, is received considerable attention. Semantic relations between two words extracted automatically from text data are in such type of data. They are widely available for many tasks of natural language processing (NLP) applications, such as word-sense disambiguation \cite{2}, paraphrasing \cite{3}, document summarization \cite{4} and machine translation \cite{5}. For instance, if we chat with robots and they cannot extract semantic relations from our words, the conversation does not smoothly proceed. Thus, an efficient semantic relation classification mechanism is required to obtain the background knowledge of robots \cite{6,7}, so that data can be applied more smartly on existing intelligent systems.

In the past few years, relation classification has attracted considerable research interest. Many approaches have studied relation classification, the most representative and general one is that of
supervised classification from labeled data, which has been shown to be reliable and yield good classification results in most cases [8–11]. In supervised classification, feature-based approaches are used most commonly. In these approaches, to achieve high-level accuracy, a set of heuristic features that can effectively represent relations between two nominals must be determined. Since using richer and higher-quality features leads to a better performance, in existing approaches, frequently various features, such as part-of-speech (POS) tagging, syntactic patterns, and prepositions [12, 13], are used and external resources, such as WordNet data, Wikipedia data, and Google n-grams [14], are imported. This tendency also increases the complexity of the feature set which resulting in an increased processing time. Thus, it is difficult to simplify the complexity of the features and improve classification results simultaneously.

To solve these problems, there are two techniques that have received considerable attention: distributed representations for words and neural network language models (NNLMs). In general, an appropriate set of vectors in \( \mathbb{R}^N \), where words are mapped and \( N \) is sufficiently small, e.g., less than several hundreds, and the elements of which are usually not zero, is called a distributed representation of words [15]. Zeng et al. (2014) [16] showed that using the convolutional deep neural network (CDNN) model with lexical and sentence-level features yields better results than other existing approaches. In the CDNN model, as well as other NNLMs, words must be converted into vectors through some distributed representation before the model is applied.

An appropriate distributed representation is the key point for achieving highly accurate classification results. Most of the existing methods are combined with several feature sets to construct the appropriate distributed representation. Even some feature sets are extracted from external resources. Table I shows number of external resources and feature sets in the existing research. Although the use external sources can improve classification accuracy. It has obvious drawbacks; it makes the algorithm of approach more complex, and increase the number of dimension of the feature vectors. If the object language has been changed, the external resources used in the approach will be invalidated. Our motivation is to find a simple way to build a simple and small feature set, and to use only that feature set to obtain high classification accuracy. In other words, we need a simple and smart representation of data that contains a considerable amount of potential syntactic and semantic information.

| System Name       | # of Feature Sets | # of External Resources |
|-------------------|-------------------|-------------------------|
| Baseline          | 1                 | 0                       |
| ECNU-SR-7         | 5                 | 2                       |
| ISI               | 4                 | 3                       |
| FBK_IRST_12VBCA   | 4                 | 1                       |
| UTD               | 6                 | 5                       |
| RMVN              | 2                 | 1                       |
| CDNN              | 2                 | 1                       |
| CR_CDNN           | 2                 | 1                       |
| RelEmb\_FULL      | 3                 | 3                       |
| Proposed          | 1                 | 0                       |

We obtain the inspiration from the related research on word vectors. Distributed representations of words called word vectors proposed by Mikolov have shown to preserve linguistic regularities, such as the semantic relations between two words [17, 18]. For example, it is known that \( v(\text{king}) - v(\text{queen}) \approx v(\text{man}) - v(\text{woman}) \), where \( v(w) \) is the vectors of the word \( w \). Fu et al. (2014) showed that the prediction performance for the hypernym-hyponym relation is improved in terms of F1 scores by using word vectors [19]. Since word vectors appear to contain information about other relations, we exploit them in our proposed approach.

In this paper, we introduce new distributed representations for sequences of words between two words, called substring vectors. We used it as only a feature set for relation classification and achieved

29
a sufficiently high accuracy (78.10%). More importantly, different from existing approaches, our approach do not use any external resources, and has a low number of dimension for feature vectors. The novelty of our approach is that we have proposed a new simple but effective weighting method based on words frequency. The experiments showed that after processing the weighting, the F1 scores of the classification results can be improved by 1%–3%.

This paper is organized as follows. First, we discuss the problems of existing researches and solutions for relation classification in Section 2. Then, we explain the details of word vectors that the origin of our ideas of the proposed method in Section 3. A detailed description of our approach is then provided in Section 4. In Section 5, We show the using external tools and dataset when implementing the proposed method. The comparison of our method to other similar methods, and the results of experiments are given in Section 6.

2. Related work

2.1 Learning with sophisticated features

Relation classification is one of the most important topics in natural language processing (NLP). A benchmark dataset for semantic relations called SemEval-2010 Task 8, which was used in a contest, picks up nine relations that cover a sufficiently broad range to be of general and practical interest [20]. The approach of Bryan et al., won the relation classification contest [14], uses various types of features, which can be partitioned into eight groups, where five groups are taken from external resources. This shows that the combination of rich features and learning algorithms that are tolerant to high dimensions, such as the linear support vector machine (SVM), is one of the most effective approaches for relation detection. However, the performance of the approach strongly depends on the quality of the designed features and the amount of external resources.

2.2 Learning with NNLMs

Bengio (2003) proposed a neural probabilistic language model [21] that is also increasingly used to solve the problems of NLP. With the recent revival of interest in deep neural networks (DNNs), many researchers have concentrated on using deep learning approaches to learn features. Socher (2012) proposed a new recursive neural network (RNN) to learn vectors for relation classification [22]. Motive by RNN, Hashimoto (2013) proposed an explicit weighting of important phrases for the target task. Their experimental results on semantic relation classification show that weighting significantly improves the prediction accuracy of the model [23]. Recent research on relation classification has used a CDNN to extract lexical and sentence level features [16]. Based on CDNN, Sanotos (2015) [24] proposed a pair-wise ranking loss function that makes it easy to reduce the impact of artificial classes. This approach have achieved state-of-the-art results for relation classification.

The above studies showed that using NNLMs improves results more than other existing approaches, and appropriate weighting of features improves the quality of vectors. Unfortunately, because of the large number of dimensions, it is difficult for existing approaches to reduce the computational cost while maintaining prediction accuracy. To solve this problem, we propose low-dimensional feature vectors for relation classification. Our approach effectively alleviates the shortcomings of traditional features as described in the next section.

2.3 Comparison with existing approaches

In order to express more clearly the novelty and advantage of our work, we compared our proposal with existing approaches from two aspects: one is the novelty and the advantage of our algorithm, and the other one is the simplicity and its advantage as a consequence. We mention here two methods as resent NNLM methods of semantic classification for comparison; Zeng’s method [16], which is representative for others using CNNs [19], and Hashimoto’s method [25], which is most similar with our method to the best of our knowledge.

First, we describe the novelty and the advantage of our method compared to others. The clearest novelty of our method is that we give a new weighting method to make feature vectors. All method including our method generally divides a sentence into three pieces at first, which are the inside
word sequences and the word sequences on the left and the right hand side (i.e., $\text{Substr}_2$, $\text{Substr}_1$ and $\text{Substr}_3$ in Section 4.3). In Hashimoto's method, a vector representation for each inside word are learned by their proposed probabilistic model, where is similar with CBOW but specialized to represent the above segmentation of each sentence. Then, to make a feature vector, they take an arithmetic mean of all inside word vectors with a certain length of the window. They classify semantic relations by feeding feature vectors to the single-layered softmax classifier. On the other hand, Zeng’s method first makes a word vector for each inside word by the well-known embedding methods such as CBOW, and then feeds the sequence of those vectors along with other sophisticated features to the CNN.

Our method is relatively more similar with Hashimoto’s method than Zeng’s CNN-based method, in terms that both our and their methods averages word vectors corresponding to the inside word sequences. However, our method makes a final feature vector by the specific weighted arithmetic mean, which is one of the main novelties of our method, whereas Hashimoto’s method uses the simple arithmetic mean.

As shown in Section 6.4, our weighting method improves the result by 1%–3% comparing to the method using the simple arithmetic mean. This improvement is sufficiently considerable comparing to other methods. For example, the specialized word vectors proposed by Hashimoto, which is one of the main novel point of their methods, improves by 1% comparing to the method replacing their specialized word vectors with usual ones obtained by well-known methods like CBOW. In addition, our weighting method is able to be combined with other methods that uses the simple arithmetic mean of word sequences like Hashimoto’s.

In the following, we describe the simplicity and its advantage of our approach. One of the main simplicity of our approach is that it has significantly small number of dimensions of input feature vectors, which we call the length of input vectors below. In Zeng’s method, the length of input vectors is at least $d(3l + 8)$, where $d$ is the number of dimensions of each word vectors and $l$ is the variable length of the inside word sequence. Since they let $d = 50$ in their paper and $l$ is approximately 17 in average for the used data, the length of input vectors is no less than 2950 approximately. In Hashimoto’s method, the length of input vectors is at least $4d(2 + c)$ as they describe, where $c$ is the length of the window each word has. By plugging $d = 100$ and $c = 3$ which are the values they found by tuning, we have that the length of input vectors for their method is approximately 2000. In contrast to those methods, our approach makes input vectors with the length of $d$, and we find that the results are best when $d = 50$ approximately.

The significantly small length of the input vectors has some obvious advantages. One of them is the low computational cost for both learning and predicting. Unfortunately, we could not find the actual computing time for both Zeng’s and Hashimoto’s methods. However, if a classifier is fixed, its computational cost for learning and prediction significantly depends on the length of input vectors in many cases. In fact, the experiments shown in Section 6.3 demonstrates that general claim at least for two specific classifiers.

We also point out that no external resources are mandatory in our approach, whereas at least one external resources are used in almost all the existing researches as Table I shows. Generally speaking, non-external-resource approaches reduce the human costs and increase the robustness when applying them to new languages and domains. The experiments show that our approach still achieved a sufficiently high accuracy even though it is non-external-resource and has simple low dimensional feature vectors.

3. Distributed representations of words

Distributed representations of words in a vector space help learning algorithms to achieve a better performance in NNLM processing tasks, which is also called word embedding or word vectors [26, 27]. This idea has been applied to statistical language modeling with considerable success [21]. Mikolov et al. proposed two new language models called the continuous bag-of-words (CBOW) model and the skip-gram model, both of which are a type of unsupervised NNLM [17, 18]. They also proposed an
approach called negative sampling, where negative examples that do not exist in data are constructed in order to reduce the problem to an optimization in a discriminative model [28].

In the CBOW model with the negative-sampling approach, a center objective function parameterized by input and output vectors of words is optimized.

Intuitively, this optimization makes the vector of a word \( w \) as close as possible to the average of the vectors of words in the context of \( w \) with the length of \( k \), while each vector repel each other due to the effect of negative samples. The word vectors obtained above are known to produce good representations that reflect semantics of words.

For relational classifications, since there are multiple words between two objective words, the word-vector representations are still redundant to be considered as input vectors for existing classifiers. Thus we define and construct new efficient vectors for substrings based on word vectors as described in the following section.

4. Proposed method

In this section, we propose a new approach for classification of relations between pairs of nominals. We introduce a simple but effective feature vector called substring vector, which is constructed above word vectors. The process for creating the substring vectors consists of three steps. Firstly, a vocabulary is constructed from the training text data and then vector representations of words are learned. Secondly, substrings between pairs of nominals in sentences are extracted and a weight for each word vector is calculated. Finally, after the vectors for each word in a substring are weighted and normalized, the sum of the vectors constitutes the substring vectors.

For instance, in the sentence “This chair is made of wood,” the words “chair” and “wood” have the Entity-Origin relation. The substring specified by these words and to be mapped into \( \mathbb{R}^N \) is “is made of.” The weight of each word in this substring is different. Words that carry more relation information should be more weighted, while ubiquitous words should not. Our approach defines the weight of each word through its frequency in the text, so that if the word frequently appears in the substring, it has a higher weight score. In the above example, the weight score of “make” and “of” is higher than “is.” According to this rule, we can obtain a weight dictionary for words. Then, we can perform weighting and normalizing for the original vector. Using the processed word vector of each word in the substring, we sum them to construct the substring vectors. For instance, \( v(substring_1) = v(is) + v(made) + v(of) \), and \( v(substring_1) \) are the corresponding substring vectors for “chair” and “wood.” The substring vectors are a kind of feature for training classifiers for relation classification.

In the following, we provide a detailed description of our approach.

4.1 Assumed input data

We denote \( S_1, \ldots, S_M \) be the sentences in the data and for each \( i \), \( S_i = w_{i1} \cdots w_{i|S_i|} \), where \( w_{ik} \) represents the \( k \)-th word of \( S_i \). We denote the set of all sentences \( D = (S_1, \ldots, S_M) \). We assume that each sentence \( S_i \) have at most one pair of indices \( e_{i1} \) and \( e_{i2} \), where \( w_{i_{e_{i1}}} \) and \( w_{i_{e_{i2}}} \) are the pair of words to be classified with respect to semantic relations. To escape double subscripts, we let \( w(e_{ik}) \) denote \( w_{i_{e_{i_k}}} \) for \( k = 1, 2 \). If \( S_i \) have no such pair of words, we let \( e_{i1} = e_{i2} = 0 \) and \( w(0) = \lambda \), where \( \lambda \) denotes the empty word. Input data is represented by the sequence of triplets \( (S_1, e_{11}, e_{12}), \ldots, (S_M, e_{M1}, e_{M2}) \). We allow \( e_{i1} \) and \( e_{i2} \) to be chosen arbitrarily. For instance, every pair of locations of nominals except for stop words appeared in a sentence can be regarded as \( e_{i1} \) and \( e_{i2} \). For another instance, if we use data sets like SemEval-2010 Task 8, since each pair of nominals to be classified is marked as indices from the beginning for each sentence \( S_i \), we can use them as \( e_{i1} \) and \( e_{i2} \). It is to be noted that this assumption does not include any information about classification for those pairs, since no label except for \( e_{i1} \) and \( e_{i2} \) is given to \( S_i \).

4.2 Learning vector representation of words

Since we construct the substring vectors based on word vectors [15], the first step is to obtain word vectors for all words in \( D \). All sentences in \( D \) are used as a training data for algorithms, such as the
CBOW model with the negative sampling described in Section 3. Word vectors are in \( \mathbb{R}^N \), where \( N \) is arbitrarily chosen but is usually around 10–100. Note that, in this step, although, for each sentence, we have the sequence of word vectors which we believe that have potentially enough information about semantic relations, it is still required to introduce another distributed representation for the sentence itself as described later. Simple concatenation of the word vectors in each sentence is too naive and has too large dimension to be classified well.

4.3 Extraction of substrings from sentences
Each sentence \( S_i \) is divided into three substrings by splitting \( S_i \) at \( e_{i1} \) and \( e_{i2} \). We denote \( \text{Substr}_{i1} \), \( \text{Substr}_{i2} \) and \( \text{Substr}_{i3} \) be those substrings in order. The substring that we mainly map into the vector space is \( \text{Substr}_{i2} \), since \( \text{Substr}_{i2} \) is most informative about the semantic relation between \( w(e_{i1}) \) and \( w(e_{i2}) \). For instance, we suppose that the data set \( D \) is consisted of only one sentence i.e., \( D = (S_1) \), where

\[
S_1 = \text{The}_1 \text{ eye}_2 \text{ works}_3 \text{ using}_4 \text{ the}_5 \text{ retina}_6 \text{ as}_7 \text{ a}_8 \text{ lens}_9 . ,
\]

\( e_{i1} = 2 \) and \( e_{i2} = 6 \), i.e., \( w(e_{i1}) = \text{“eye”} \) and \( w(e_{i2}) = \text{“retina”} \). We denote \( w_i \) be the \( i \)-th word of \( S_1 \). \( S_1 \) can be thought as a sentence that describes the relations between \( e_{i1} \) and \( e_{i2} \). Then, we have

\[
\begin{align*}
\text{Substr}_{i1} &= (w_{11}) = (\text{“the”}), \\
\text{Substr}_{i2} &= (w_{13}, w_{14}, w_{15}) = (\text{“works”}, \text{“using”}, \text{“the”}), \\
\text{Substr}_{i3} &= (w_{17}, w_{18}, w_{19}) = (\text{“as”}, \text{“a”}, \text{“lens”}),
\end{align*}
\]

and \( \text{Substr}_{i2} \) looks to explain the relation between “eye” and “retina”, and are best among the three. Figure 1 shows sentences divided by \( e_{i1} \) and \( e_{i2} \) in our approach. By putting the substring in each sentence of text data together, we can get three multisets of substrings: \( \text{Set}_{i1} \), \( \text{Set}_{i2} \) and \( \text{Set}_{i3} \). \( e_{i1} \) and \( e_{i2} \) might appear continuously in one sentence, that means exist sentence without \( \text{Substr}_{i2} \). This sentence will be treated as invalid data, and not within the scope of our approach. But \( \text{Substr}_{i1} \) and \( \text{Substr}_{i3} \) as the empty is allowed.

![Fig. 1. Sentence \( S_1 \) is divided into three parts, \( \text{Substr}_{i1} \), \( \text{Substr}_{i2} \), and \( \text{Substr}_{i3} \). \( \text{Set}_{i1} \), \( \text{Set}_{i2} \), and \( \text{Set}_{i3} \) are multisets of substrings.](image)

4.4 Constructing distributed representation of substrings
After word vectors are learnt and all sentences are divided into substrings, we make substring vectors. The process of construction is shown in Fig. 2. First, we create the weight dictionary based on word frequencies. Then, we use this dictionary for weighting and normalization of each words in substrings, and obtain substring vectors by averaging them.
4.4.1 Normalized weight of words in substrings

To construct a reasonable representation for a substring from word vectors, we define weights for each word which represent degrees of importance of relations between pairs of nominals. For instance, suppose that a data set \( D \) includes only one sentence \( S_1 \), i.e., \( D = (S_1) \). Among “works”, “using” and “the”, which are elements of \( \text{Substr}_{ij} \), “the” appears in both \( \text{Substr}_{1i} \) and \( \text{Substr}_{12} \) while “works” and “using” appear only in \( \text{Substr}_{12} \). Thus, we observe that, compared to “the”, “works” and “using” are more informative for the semantic relation between “eye” and “retina.” In other words, if a word \( w \) appears mainly in \( \text{Set}_2 \), we believe that \( w \) often plays a role of representing some semantic relation.

For a word \( w \) and a substring \( s \), we define \( \text{Cnt}(w, s) \) as the number of occurrences of \( w \) in \( s \). In addition, for a multiset of substrings \( \mathcal{S} \), we denote \( \text{Cnt}(w, \mathcal{S}) = \sum_{s \in \mathcal{S}} \text{Cnt}(w, s) \). The weight for a word \( w \) is defined as follows:

\[
    a(w) = \frac{\text{Cnt}(w, \text{Set}_2)}{\text{Cnt}(w, \text{Set}_1) + \text{Cnt}(w, \text{Set}_2) + \text{Cnt}(w, \text{Set}_3)}.
\]

In order to prevent that the length of each substring vector is too large or too small, or to have the center of gravity for weighted word vectors, we normalize the weights in Eq. (1) with respect to each substring \( \text{Substr}_{ij} \):

\[
    \pi_{ij} = \frac{a(w_{ij})}{\sum_{k=e_{i1}+1}^{e_{i2}-1} a(w_{ik})}.
\]

4.4.2 Vector representation of substring

Using the normalized weights and the word vectors, we define the substring vectors \( v(\text{Substr}_{ij}) \) for each substring \( \text{Substr}_{ij} \) as follows:

\[
    v(\text{Substr}_{ij}) = \sum_{k=e_{i1}+1}^{e_{i2}-1} \pi_{ij} v(w_{ik}).
\]

where \( v(w) \) is the word vector of \( w \), shown in Fig. 2.

In many language models, word vectors of a given sentence or substring are simply concatenated and treated as an input vector in their lowest layer. However, since this approach increases the dimension of input vectors, it courses high computational cost. Furthermore, since the number of words in a sentence or substring can be changed, it is also difficult to unify the number of dimensions. Based on the above two points, our approach proposes a simple way to make the substring vectors.

In previous researches, it is known that making input vectors including external knowledge such as WordNet increases the prediction accuracy. In this paper, to verify effectiveness of our approach, we use only substring vectors as input vectors for classifiers.
4.5 Multiclass classifiers

In recent research, the dimension of feature vectors tends to be extremely large since including much information as far as effective is believed to advocate high accuracy classification. In those cases, leaning algorithms must have simple structures such as linear and at most small-order polynomial models due to the problems of computational cost and overfitting. On the other hand, in our approach, we focus on exploring a future space which is enough low-dimensional and smart so that words are mapped keeping their semantic relations. In stead, we use rather non-linear and flexible classifier such as SVM with gaussian kernels, where it is hard to apply to extremely high-dimensional data sets.

5. Implementation

We implemented the proposed method in Java\textsuperscript{1}, and used two external open-source softwares. In semantic relation classification, we choose the same text data as other similar researches, which includes two parts, train and test datasets. We calculated their substring vectors as feature vectors for each part, and applied classifiers to obtain classification results.

5.1 Dataset

To evaluate the performance of our proposed method, we used the SemEval-2010 Task 8 dataset \cite{20}. The dataset is freely available and contains 10,717 annotated examples, including 8,000 training instances and 2,717 test instances. It distinguishes nine semantic directed relations, such as Entity-Origin, Component-Whole, and Cause-Effect. In addition, it has another special undirected relation called Other.

In this dataset, as shown in Fig. 3, each training sentence has one pair of nominals tagged with \texttt{<e1>} and \texttt{<e2>} and specifies one semantic relation. Note that semantic relations have a direction. For instance, although both (2) and (1) in Fig. 3 have the relationship called Member-Collection, these two instances cannot be classified into the same category when we attempt to learn, because Member-Collection (e2,e1) and Member-Collection (e1,e2) should be distinguished.

| (1)   | “It describes a method for loading a horizontal \texttt{<e1>stack</e1> of containers into a \texttt{<e2>carton</e2>.”} \hspace{1cm} : Entity-Destination (e1,e2) |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (2)   | “There is a \texttt{<e1>nest</e1>} of \texttt{<e2>rabbits</e2>} up in the loft.” \hspace{1cm} : Member-Collection (e2,e1) |
| (3)   | “This \texttt{<e1>surgeon</e1>} is part of the study \texttt{<e2>group</e2>}.” \hspace{1cm} : Member-Collection (e1,e2) |

Fig. 3. Examples of sentences and their semantic relations labels.

5.2 External tools and hyperparameters

We used two external software programs for the implementation. The first one is called word2vec\textsuperscript{2} and is for learning word vectors from sentences based on the CBOW model. Before learning the word vector, we had to clean the data; for instance, we removed tags in given sentences, unnecessary symbols, and substrings in parentheses. The parameters for word2vec were set as follows. We set the embedding dimensionality to \{20, 40, 60, 80, 100, 200, 300, 400, 500\}, the window = 5, sample = \(1e^{-4}\), \(iter = 15\), \(cbow = 1\), and the learning rate be the default value. We extracted a corpus from the SemEval-2010 Task 8 dataset. It includes 10717 sentences and 184,877 words (1.1 MB). Then the word2vec was trained by this corpus.

\textsuperscript{1}We published source code of proposed method on GitHub repository: https://github.com/kinten0902/Task8

\textsuperscript{2}https://code.google.com/p/word2vec

35
The second software is called Weka\(^3\) and is for the classification. After the substring vectors as feature vectors were constructed, we applied three multi-class classifiers for them: random forest (RF), SVM with the Gaussian radial basis function kernel (SVM-RBF), and the linear SVM (SVM-LN). We set almost all the hyperparameters as default, except for the number of trees for RF, and the penalty term for incorrect Classification of SVM-RBF. Those two hyperparameters are searched by cross validation. As a result, for the parameter settings of RF, we let the number of trees be 120. For those of SVMs, we let the penalty term for incorrect Classification of SVM-RBF be 60. In Appendix A, We show the sensitivity of hyperparameters and how we determined them.

6. Experiments

In this section, we conducted five sets of experiments for evaluating our approach. The results of experiments demonstrated that our approach yielded better classification results (F1 score: 77.18\% and 78.10\%) than most of sophisticated features approaches, however there was still a gap in the state-of-the-art methods which used NNLMs (almost all F1 score above 80\%). Crucially, unlike other existing approaches, our method did not use any external resources and the feature vectors had a sufficiently low-dimensional (40–60). In addition, we verified the validity of the proposed weighting method and dimension reduction method through experiments.

6.1 Measure of evaluation

We learned from the training data and obtained F1 scores from the test data for 10 relations including Other. The average of the F1 scores for nine relations excluding Other is called the macro-averaged F1 score. In addition, we removed the instances of Other from the training and test data and obtained the average of the F1 scores for nine relations, which is called micro-averaged F1 score. To compare results of our proposed method in Section 6.2, we adopted the macro-averaged F1 score as a measure of the prediction accuracy that is same as previous studies. However, we adopted the micro-averaged F1 score in other experiments (Section 6.3–Section 6.6) because the classification results will be more stable when excluding occurrences of Other.

6.2 Comparison of the proposed method with existing methods

To evaluate the performance of our proposed method, we compared six methods with our method, as shown in Table II. The first four were the best of the existing approaches that are not NNLMs and the following two were proposed in current studies using NNLMs. The Table III shows F-scores as prediction accuracies for training sets consisting of 1000, 2000, 4000, and 8000 sentences (TD1–TD4). Since the initial vectors of word2vec are random, the learnt vectors have also a certain range of changes. Thus, we took the average values of 10 executions to avoid instability in our experimental results. In addition, to compare our results with those obtained in previous studies, we adopted the macro-averaged F1 score as a measure of prediction accuracy.

The F-scores of our proposed methods used the weighted method, and used CBOW model for learning the word vector representations. At the beginning of experiments, we used two models, the CBOW and the Skip-gram. However we found that the performance of CBOW was better than Skip-gram in F-scores (0.9\%–1.5\%). The number of dimensions of RF is 60 and SVM-RBF is 40. The results were obtained without the using of PCA and ICA \cite{30, 31}.

As shown in Table III, in spite of the very simple feature set, the results of the proposed methods were comparable with other methods in terms of prediction accuracy. The table shown that the existing approaches use various kind of features, including external data sources, to produce comparable results. While NNLM methods (RMVM and CDNN) deviated from this tendency of existing approaches, they still use external data sources, such as WordNet. The proposed method used only one kind of feature without any external data source. As a result, the computational cost for learning and classifying is also sufficiently small. The proposed method appeared to successfully achieve relatively simple features, small computational cost, and high classification accuracy simultaneously.

\(^3\)http://www.cs.waikato.ac.nz/ml/weka
Table II. Classifier, feature sets, and resources used for relation classification.

| System             | Classifier | Feature sets                                                      | Resource used                  |
|--------------------|------------|-------------------------------------------------------------------|--------------------------------|
| Baseline [20]      | Naive Bayes| local context of 2 words only                                      | -                              |
| ECNU-SR-7 [12]     | SVM        | stem, POS, syntactic patterns, hyponymy and meronymy relations    | Word Net and syntax            |
| ISI [29]           | Maximum Entropy | a noun compound relation system, various feature related  | Word Net, Roget's Thesaurus  |
|                    |            | to capitalization, affixes and closed-class words                 | and Google n-grams             |
| FBK_IRST_12VBCA [13]| SVM        | 3-word window context features (word form, part of speech, or   | Cyc                            |
|                    |            | orthography) + Cyc; parameter estimation by optimization on     |                                 |
|                    |            | training set and verbs                                           |                                 |
| UTD [14]           | SVM and   | context words, hypernyms, POS, dependencies, distance, semantic   | WordNet, syntax, Google       |
|                    | two-step  | roles, Levin classes, para-phrases                               | n-grams, PropBank/NomBank      |
|                    | classification |                                                                     | and Levin classes              |
| RMVM [22]          | MVRNN      | POS and NER                                                       | WordNet                        |
| CDNN [16]          | softmax    | word pairs, words around word pairs                               | WordNet                        |
| CR_CDNN [24]       | softmax    | word embeddings, word position embeddings                        | Wikipedia                      |
| RelEmbFULL [25]    | softmax    | embeddings, dependency paths, WordNet, NE                       | WordNet, Named Entity tags,    |
| Proposed           | RF, SVM-RBF| substring vectors                                                 | Wikipedia                      |

6.3 Comparison of scores and computing time in different dimensions
Figures 4 and 5 shown the F1 scores and computing time for the first set of experiments. If the dimension of the substring vectors is less than approximately 400, the nonlinear classifiers obtained better classification results than the linear one. However, when the dimension is greater than approximately 400, the linear classifier had better performance than the others. This phenomenon appears to be due to the overfitting caused by the high degree of freedom that the SVM-RBF is as a classifier. This implied that a sufficiently low-dimensional representation of the data is required in order to use nonlinear classifiers efficiently. In the second experiment, we obtained the best classification results using nonlinear classifiers for approximately 50 dimensions of the substring vectors, and the RF obtained better performance than the other classifiers with respect to the computing time.

6.4 Effect of the proposed weighting method
In this set of experiment, as shown in Fig. 6, after processing the weighting, the F1 scores of the classification results can be improved by 1%–3%. We ensured that our weighting method is effective by comparing the weighted and non-weighted substring vectors as feature vectors of RF. This is because words that appear frequently in substrings between pairs of nominals to be classified ($e_{i1}$ and $e_{i2}$) are expected to carry much information about semantic relations. Besides, we have tried three other kinds of weighting methods as shown in Table IV. Our approach without weighting to be used as a baseline, and W4 represents the weighting method we adopted in our approach as described.
Table III. F1 score of all systems for the test dataset as a function of training data: TD1=1000, TD2=2000, TD3=4000, and TD4=8000 training examples.

| System          | TD1 | TD2 | TD3 | TD4 | Best Cat | Worst Cat |
|-----------------|-----|-----|-----|-----|----------|-----------|
| Baseline        | 33.04 | 42.41 | 50.89 | 57.52 | MC (75.1) | IA (28.0) |
| ECNU-SR-7       | 58.67 | 58.87 | 72.79 | 75.21 | CE (86.1) | IA (61.8) |
| ISI             | 66.68 | 71.01 | 75.51 | 77.57 | CE (87.6) | IA (61.5) |
| FBK_IRST_12VBCA | 63.61 | 70.20 | 73.40 | 77.62 | ED (86.5) | IA (67.3) |
| UTD             | 73.08 | 77.02 | 79.93 | 82.19 | CE (89.6) | IA (68.5) |
| RMVM            | -    | -    | -    | 82.4 | -        | -         |
| CDNN            | -    | -    | -    | 82.7 | -        | -         |
| CR_CDNN         | -    | -    | -    | 84.1 | -        | -         |
| RelEmbFULL      | -    | -    | -    | 83.5 | -        | -         |
| Proposed (RF)   | 69.17 | 72.27 | 75.07 | 77.18 | CE (92.50) | CW (67.00) |
| Proposed (SVM-RBF) | 70.84 | 73.11 | 76.38 | 78.10 | CE (92.90) | CW (65.70) |

Fig. 4. F1 scores for each classifier as a function of the dimension of the substring vectors.

In Section 4.4, in Appendix B, we shown the detail of algorithms for W1, W2 and W3. Through the experimental results, we can see that the method 1, 2 and 3 shows the worst performance. We considered the case where there are not enough training data to obtain the weights. Therefore, the weighting method 4 is most effective from a small amount of learning data.

6.5 Number of dimensions and degree of freedom for classifiers

The results of the fourth set of experiments are shown in Fig. 7. We ensured that overfitting occurs more easily for a higher degree of freedom for the classifiers, which is a similar result obtained in the first set of experiments. We had the best combination when the degree of the kernel function is not one and the number of dimensions is approximately 40–60.
Fig. 5. Computing time for each classifier as a function of the dimension of the substring vectors.

Fig. 6. Comparison of the effect of substring vectors (weighted average) with the simple average of word vectors as feature vectors. In this experiment, we measure scores using 10-fold cross-validation (CV) in the training data for stability.

Table IV. F1 score of weighting methods ($d = 40$).

| Weighting Method | F  |
|------------------|----|
| NonW (baseline)  | 79.73 |
| W1               | 51.4 |
| W2               | 50.9 |
| W3               | 52.1 |
| W4               | 81.28 |

6.6 Effect of dimension reduction using PCA and ICA

The purpose of the last two sets of experiments is to examine whether the decrease of the accuracy is caused by the classifiers or the word embedding methods. After we constructed word vectors in 500 dimensions, we reduced the dimensions of the word vectors by PCA and ICA. The eigenvalues
Fig. 7. F1 scores for each degree of the polynomial function kernel SVM as a function of the dimension of the substring vectors.

Fig. 8. Eigenvalues as a function of the number of dimensions for the PCA of the original word vectors.

for PCA are shown in Fig. 8. The figure shown that almost all the word vectors are scattered approximately in a 100 dimensional subspace. We extracted several groups of reconstructed vectors in different dimensions. The classification results of each group are shown in Fig. 9. The F1 score of our method used word vectors in 50 dimensions with PCA is almost same or a little lower than that in 50 dimensions without PCA. The F1 score of the original 500 dimensions is 0.811 as a baseline for comparison. If the dimension of reconstructed vectors is less than 300, the processed data obtained better classification results than the original 500 dimensions. When the dimension is greater than 300, the F1 score decrease. We observed that even if PCA does not reduce the number of dimensions, that is, when the dimension after PCA is still 500, the score get worse than the original one. This is because PCA transforms the coordinate and changes the distances among embedded words. We also observed that whichever PCA or the word-embedding method we use for reducing the number of dimensions, the score is maximized at 40–50. That fact indicated that the decrease of the accuracy is mainly caused by classifiers or the curse of dimensions.
7. Conclusion
In this paper, we proposed a new distributed representations—substring vector, and used it as a feature set for relation classification. Through this simple vector representation, we successfully extracted information about semantic relations between pairs of nominals. With almost no optimizing the parameters of the classifier and without using any external resources, our approach yielded comparable classification results with most of sophisticated features approaches. We also shown that the proposed weighting method of substring vectors can improve the results of relation classification by 1%–3% compared to the non-weighting method. Although we used nonlinear classifiers such as an RF and SVM-RBF, we hope that it may be improved by using other recent learning algorithms used in NNLMs. In addition, the length of the input feature vectors is significantly small compared to that of other existing methods. For instance, when we extract semantic relations from massive amount of unlabeled text data, it is preferable that the size of the set of features is sufficiently small so that data is processed in low-computational cost. The substring vector is applicable as an effective feature and is able to be combined with existing features.

Appendix

A. Determination of hyperparameters
We show that the adjustment of the random forest classifier parameters has a strong influence on the classification results shown in Fig. A-1. We do not optimize parameters except the number of trees in the case of RF. If the parameters of the classifiers are fully optimized, we may achieve results that are a little better than those presented in this paper. We observed that a similar phenomenon occurs on RBF classifier when adjusting the penalty parameter.

B. Various weighting methods
We explain the details of weighting methods W1, W2 and W3 in Section 6.4. In each weighting method, we define a weight for each word $w$ in each sentence $S_w \in C$, where $C$ is a given corpus. In all weighing methods W1–W4, generally speaking, we basically reduce the weights of the general or frequent words. The weight $a_1$ for the word $w$ is defined as: $a_1(w) = \log(|C|/\text{Cnt}(w, C_{\text{other}}))$, where $|C|$ is the total number of sentences in $C$ and $\text{Cnt}(w, C_{\text{other}})$ is the occurrence number of $w$ that appears in $C_{\text{other}}$. The intuitive meaning of W1 is that in corpus $C$, if the word $w$ frequently appears in other sentences except for $S_w$, i.e., in $C_{\text{other}} = C - \{S_w\}$, we reduce its weight. It should be noted that the method W1 has approximately the same weights as the tf-idf if we consider that each sentence is a separated document. The method W2 is defined similarly with W1 except
Fig. A-1. F1 score and learning time of RF as a function of a parameter of RF (the number of learnt trees). In this experiment, we measure scores using 10-fold cross-validation (CV) in the training data for stability.

Fig. B-1. The definitions of the $S_w$, $C_{other}$ and $M_{other}$.

that we count the word $w$ only in $M_{other}$, where $M_{other}$ represents the common part of Set2 and $C_{other}$. We show the definitions of the above notations in Fig. B-1. The weight $a_2$ in W2 is defined as: $a_2(w) = \log(|C|/\text{Cnt}(w, M_{other}))$. In W3, the weight of the word $w$ depends on the target labels of the semantic relations of $S_w$. We divide $C$ into $\{C_{\text{label1}}, ..., C_{\text{label9}}\}$ according to their target label of semantic relation. In the method W3, we define the weight $a_3$ of the word $w$ as: $a_3(w) = |C|/\text{Cnt}(w, C_{\text{labeli}})$.

References
[1] Z. Jin, C. Shibata, and K. Tago, “Relation classification through substring representations,” In Proceedings of the 2015 International Symposium on. Nonlinear Theory and its Applications (NOLTA2015), 2015.
[2] J.-W. Fan and C. Friedman, “Word sense disambiguation via semantic type classification,” In AMIA Annual Symposium Proceedings, p. 177, 2008.
P. Nulty and F. Costello, “General and specific paraphrases of semantic relations between nouns,” *Natural Language Engineering*, vol. 19, pp. 357–384, 2013.

D. Radev, J. Otterbacher, and Z. Zhang, “Cross-document relationship classification for text summarization,” *Association for Computational Linguistics*, 2008.

W.Y. Zou, R. Socher, D.M. Cer, and C.D. Manning, “Bilingual word embeddings for phrase-based machine translation,” In *EMNLP*, pp. 1393–1398, 2013.

A. Augello, G. Pilato, O. Gambino, R. Pirrone, S. Gaglio, and V. Cannella, “An Emotional Talking Head for a Humoristic Chatbot,” INTECH Open Access Publisher, 2011.

D. Zelenko, C. Aone, and A. Richardella, “Kernel methods for relation extraction,” *J. Mach. Learn. Res.*, vol. 3, pp. 1083–1106, 2003.

Y. Chen, M. Lan, J. Su, Z.M. Zhou, and Y. Xu, “Ecnu: Effective semantic relations classification without complicated features or multiple external corpora,” In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 226–229, 2010.

K. Tymoshenko and C. Giuliano, “Fbk-irst: Semantic relation extraction using cyc,” In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 214–217, 2010.

R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu, “Learning semantic hierarchies via word embeddings,” In *Proceedings of the 52nd Annual Meeting of the ACL*, vol. 1, pp. 1199–1209, 2014.

I. Hendrickx, S.N. Kim, Z. Kozareva, P. Nakov, D.Ó. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, “Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals,” In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 33–38, 2010.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

R. Socher, B. Huval, C.D. Manning, and A.Y. Ng, “Semantic compositionality through recursive matrix-vector spaces,” In *Proceedings of the 2012 Conference on EMNLP*, pp. 1201–1211, 2012.
neural networks for semantic relation classification,” In Proceedings of the 2013 Conference on EMNLP, pp. 1372–1376, 2013.

[24] C. Nogueira dos Santos, B. Xiang, and B. Zhou, “Classifying relations by ranking with convolutional neural networks,” CoRR, abs/1504.06580, 2015.

[25] K. Hashimoto, P. Stenetorp, M. Miwa, and Y. Tsuruoka, “Task-oriented learning of word embeddings for semantic relation classification,” CoRR, abs/1503.00095, 2015.

[26] G.E. Hinton, “Learning distributed representations of concepts,” In Proceedings of the Eighth Annual Conference of the Cognitive Science Society, pp. 1–12, 1986.

[27] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” In Proceedings of the 48th Annual Meeting of the ACL, pp. 384–394, 2010.

[28] M.U. Gutmann and A. Hyvärinen, “Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics,” J. Mach. Learn. Res., pp. 307–361, 2012.

[29] S. Tratz and E. Hovy, “Isi: Automatic classification of relations between nominals using a maximum entropy classifier,” In Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 222–225, 2010.

[30] A. Hyvärinen, P.O. Hoyer, and M. Inki, “Topographic independent component analysis,” Neural computation, vol. 13, pp. 1527–1558, 2001.

[31] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, “Building high-level features using large scale unsupervised learning,” In International Conference in Machine Learning, 2012.