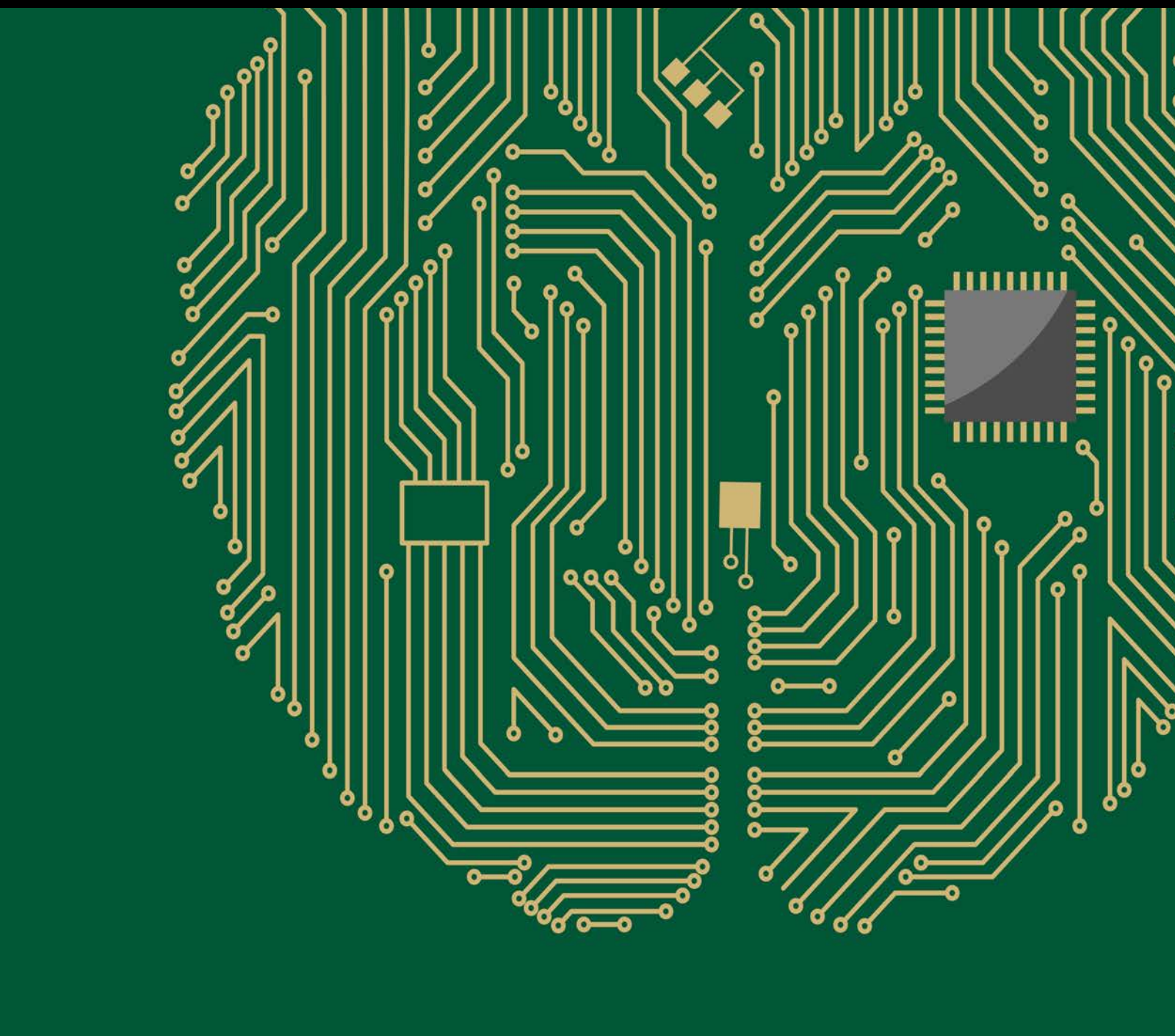
Computational Intelligence and Neuroscience

Smart Data: Where the Big Data Meets the Semantics

Guest Editors: Trong H. Duong, Hong Q. Nguyen, and Geun S. Jo



Smart Data: Where the Big Data Meets the Semantics

Smart Data: Where the Big Data Meets the Semantics

Guest Editors: Trong H. Duong, Hong Q. Nguyen, and Geun S. Jo

Copyright © 2017 Hindawi Publishing Corporation. All rights reserved.

This is a special issue published in "Computational Intelligence and Neuroscience." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board

Ricardo Aler, Spain
Pietro Aricò, Italy
Hasan Ayaz, USA
Sylvain Baillet, Canada
Theodore W. Berger, USA
Steven L. Bressler, USA
Vince D. Calhoun, USA
Francesco Camastra, Italy
Ke Chen, UK
Michela Chiappalone, Italy
Andrzej Cichocki, Japan
J. Christian Claussen, Germany
Silvia Conforto, Italy
Justin Dauwels, Singapore
Artur S. d'Avila Garcez, UK
Christian W. Dawson, UK
Paolo Del Giudice, Italy
Thomas DeMarse, USA
Piotr Franaszczuk, USA
Leonardo Franco, Spain
Doron Friedman, Israel
Samanwoy Ghosh-Dastidar, USA
Juan Manuel Gorriz Saez, Spain
Manuel Graña, Spain
Rodolfo H. Guerra, Spain
Christoph Guger, Austria
Stefan Haufe, Germany
Dominic Heger, Germany
Stephen Helms Tillery, USA
J. A. Hernández-Pérez, Mexico
Luis Javier Herrera, Spain
Etienne Hugues, USA
Pasi A. Karjalainen, Finland
Dean J. Krusienski, USA
Mikhail A. Lebedev, USA
Yuanqing Li, China
Cheng-Jian Lin, Taiwan
Ezequiel López-Rubio, Spain
Reinoud Maex, France
Kezhi Mao, Singapore
J. David Martín-Guerrero, Spain
Sergio Martinoia, Italy
Elio Masciari, Italy
Michele Migliore, Italy
Haruhiko Nishimura, Japan
Klaus Obermayer, Germany
Karim G. Oweiss, USA
Massimo Panella, Italy
Fivos Panetsos, Spain
Jagdish Patra, Australia
Sandhya Samarasinghe, New Zealand
Saeid Sanei, UK
Michael Schmuker, UK
Sergio Solinas, Italy
Stefano Squartini, Italy
Hiroshige Takeichi, Japan
Toshihisa Tanaka, Japan
Jussi Tohka, Spain
C. M. Travieso-González, Spain
Lefteri Tsoukalas, USA
Marc Van Hulle, Belgium
Pablo Varona, Spain
Meel Velliste, USA
Francois B. Vialatte, France
Ricardo Vigario, Finland
Thomas Villmann, Germany
Michal Zochowski, USA
Rodolfo Zunino, Italy

Contents

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2017, Article ID 6925138, 2 pages
http://dx.doi.org/10.1155/2017/6925138

Editorial

Smart Data: Where the Big Data Meets the Semantics

Trong H. Duong,[1] Hong Q. Nguyen,[2] and Geun S. Jo[3]

[1]*Institute of Science and Technology of Industry 4.0, Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam*
[2]*International University and Vietnam National University, Ho Chi Minh City, Vietnam*
[3]*Inha University, Incheon, Republic of Korea*

Correspondence should be addressed to Trong H. Duong; haiduongtrong@gmail.com

Received 22 November 2016; Accepted 23 November 2016; Published 26 February 2017

Copyright © 2017 Trong H. Duong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Big data technology is designed to address the challenges of the three Vs of big data, including *volume* (massive amount of data), *variety* (a range of data types and sources), and *velocity* (speed of data in and out). Big data is often captured without a specific purpose, leading to most of it being task-irrelevant data. The most important feature of data is neither the volume nor the other Vs, but its *value*. While big data is the technological foundation for data-driven business decision-making, smart data is an organized way to semantically compile, manipulate, correlate, and analyze different data sources.

To deal with the *volume*, the semantics technology facilitates better decision-making by converting massive amount of data into abstraction, meanings, and insights. Neural network algorithms offer advantages for deep learning and exploit the whole, rather than parts, of the data. The article *"A New Data Representation Based on Training Data Characteristics to Extract Drug Name Entity in Medical Text"* by M. Sadikin et al. proposes three data representation techniques to analyze the characteristics of word distribution and word similarities as a result of word-embedding training. These techniques include multilayer perceptrons, deep-network classifiers (deep belief networks, stacked denoising encoders), and long short term memory. In the article *"Objects Classification by Learning-Based Visual Saliency Model and Convolutional Neural Network"* by N. Li et al., a neuroscience-inspired classification method is proposed to simulate the human visual information processing mechanism. This method combines both visual attention model and convolutional neural network to increase the accuracy of classifying objects, especially in biology. S. Bi et al. propose a force-directed method using a fracture mechanic model to learn word embedding in the article *"Fracture Mechanics Method for Word Embedding Generation of Neural Probabilistic Linguistic Model."* The method aims to improve the accuracy, recall, and text visualization of traditional language models, and a word embedding, a semantic vector representation, could be generated via the neural linguistic model.

For the *variety*, integrating heterogeneous data sources requires effective methods for providing well-defined ontologies and natural language processing. In the article *"A Character Level Based and Word Level Based Approach for Chinese-Vietnamese Machine Translation"* by P. Tran et al., a hybrid method is proposed to translate one natural language to another (e.g., from Chinese to Vietnamese) by combining strengths of statistics-based and rule-based translation approaches at both character and word levels. In addition to using bilingual corpora, this method takes advantage of translation and word-reordering capabilities of the statistical machine translation and the translation accuracy of the rules. The approach in the article *"N-Gram-Based Text Compression"* by V. H. Nguyen et al. presents an efficient method for compressing texts (in Vietnamese) by using *n*-gram dictionaries. This approach improves compression ratio and compression and decompression times compared with other methods.

To address the *velocity*, the ontology evolution techniques support dynamically, flexibly, and adaptively creating models of new objects, concepts, and relationships and using them to better understand new cues in the data that capture rapidly

evolving events and situations. The article *"Social Media Meets Big Urban Data: A Case Study of Urban Waterlogging Analysis"* by N. Zhang et al. proposes a transfer-learning method to analyze urban waterlogging disasters in traffic operations management. It uses social media and satellite data; analyzes the correlation between severity, road networks, terrain, and precipitation; and adopts a multiview discriminant transfer-learning method to transfer knowledge among cities, as effectively applied in some cities in China and India. The article *"Automatic Construction and Global Optimization of a Multisentiment Lexicon"* by X. Yang et al. proposes an automatic construction and a global optimization framework of a multisentiment lexicon based on constraints of coordinate offsets. The method performs statistical training on a large corpus using neural network model, implements a sentiment disambiguation algorithm (based on word distribution density to distinguish the sentiment polarities in different contexts), and further integrates various human-annotated resources to learn the 10-dimensional sentiment lexicon for the optimization.

Different approaches are proposed to solve different problems in various areas. They aim to be accurate, actionable, and agile to feed smarter decision-making. Smart data harnesses the 3V-challenges and adopts semantics and neuroscience on the data to extract its *value*, the meeting point of the big data and the semantics.

Trong H. Duong
Hong Q. Nguyen
Geun S. Jo

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 2093406, 8 pages
http://dx.doi.org/10.1155/2016/2093406

Research Article

Automatic Construction and Global Optimization of a Multisentiment Lexicon

Xiaoping Yang,[1] Zhongxia Zhang,[1] Zhongqiu Zhang,[2] Yuting Mo,[1] Lianbei Li,[1] Li Yu,[1] and Peican Zhu[3]

[1]*School of Information, Renmin University of China, Beijing 100872, China*
[2]*School of Computer Science, Northeastern University, Shenyang 110819, China*
[3]*School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an 710129, China*

Correspondence should be addressed to Peican Zhu; ericcan@nwpu.edu.cn

Received 20 April 2016; Revised 13 August 2016; Accepted 20 September 2016

Academic Editor: Trong H. Duong

Copyright © 2016 Xiaoping Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Manual annotation of sentiment lexicons costs too much labor and time, and it is also difficult to get accurate quantification of emotional intensity. Besides, the excessive emphasis on one specific field has greatly limited the applicability of domain sentiment lexicons (Wang et al., 2010). This paper implements statistical training for large-scale Chinese corpus through neural network language model and proposes an automatic method of constructing a multidimensional sentiment lexicon based on constraints of coordinate offset. In order to distinguish the sentiment polarities of those words which may express either positive or negative meanings in different contexts, we further present a sentiment disambiguation algorithm to increase the flexibility of our lexicon. Lastly, we present a global optimization framework that provides a unified way to combine several human-annotated resources for learning our 10-dimensional sentiment lexicon SentiRuc. Experiments show the superior performance of SentiRuc lexicon in category labeling test, intensity labeling test, and sentiment classification tasks. It is worth mentioning that, in intensity label test, SentiRuc outperforms the second place by 21 percent.

1. Introduction

Opinion mining and sentiment analysis of online text have become a hot research area in recent years, which includes opinion summarization and sentiment classification. Most of these tasks would benefit from a high quality sentiment lexicon which could provide excellent sentiment features when no training data is available.

The primary form of sentiment lexicons is binary annotation with positive and negative labels, such as Sentiwordnet developed by Italian Information Technology Research Institute [1, 2], the Chinese general sentiment lexicon (NTUSD) [3] annotated by Taiwan University, the Chinese emotion dictionary from the Chinese Academy of Sciences, and English Xsimilarity. Multiple sentiment lexicons with assignments of the strength of sentiments are also constructed, such as the Affective Lexicon Ontology of Dalian University of Technology (DUT Ontology) [4]. To determine the word-level strength of sentiment, manual methods, supervised methods employing WordNet or other semantic resources, and unsupervised approaches based on large-scale corpus were proposed. But few works evaluated and optimized the accuracy of intensity annotation by introducing all possible linguistic heuristics.

In recent years, driven by diverse tasks in different fields, both the polarity word and its related target are included as a sentiment item. However, the application areas of such 2-tuple lexicons as ⟨polarity word, target⟩ are strictly limited to one specific field, and also the size of such lexicons could easily explode with the growth of training data, which causes the problem of sparseness of features. Massive online text makes the limitations of domain sentiment lexicons increasingly apparent, especially when the sentiment classification tasks vary in different areas. Thus, a general and adaptable lexicon is important for sentiment analysis to avoid this problem.

This paper presents a method of automatic construction and optimization of a multisentiment lexicon through statistical analysis of a massive online corpus. The main content of

this paper is as follows. First, we use neural network language model to obtain distributed representations of words from a massive online corpus (Sogou News Corpus, 3.17 GB) [5]. Second, we study the categorization of sentiments and select seed words for each category. After that, polarity words are selected and the semantic distances between polarity words and seed words are calculated with the distributed representations. The distance values are then converted into sentiment intensity through appropriate constraints. Finally, we evaluate the lexicon by combining linguistic heuristics in an optimization framework. Besides, we study sentiment tendency disambiguation method to improve the semantic description capability of our lexicon.

The remainder of this paper is organized as follows. In Section 2, we introduce some related works. The principle of automatic construction of SentiRuc is proposed in Section 3. Section 4 introduces a unified optimization framework. Experiments and evaluations are reported in Section 5. We conclude the paper in Section 6 with future researches.

2. Related Work

Many Chinese sentiment lexicons, such as NTUSD, HowNet, and DUT Affective Lexicon Ontology, are manually annotated to ensure the lexicon's coverage and effectiveness. But manual methods usually cost too much labor and time and also tend to be subjective; the coverage is also a concern. To provide more granularities, it is necessary to introduce statistical language model to automatically annotate sentiment category and intensity.

To label the sentiments, we should first study the sentiment categorization. As early as 1957, Osgood distributed human emotion to three aspects: strong and weak, good and bad, active and passive [6]. In 2012, Liu et al. presented the DUT Affective Lexicon Ontology which contains 7 sentiments: happiness, liking, anger, sadness, hate, fear, and surprise [5]. QuanChangqin constructed Ren-CECps with 8 kinds of sentiments: expectation, joy, love, surprise, anxiety, sorrow, angry, and hate [7]. But existing categorizations of emotions are asymmetric. For example, there is no opposite emotion of "surprise" or "fear," which can cause inconvenience in feature extraction and selection in supervised methods of sentiment analysis. Besides, there is coupling between emotion categories, such as "praise" and "like." Therefore, sentiment classification needs to be investigated to suit both psychology and computational linguistics.

In addition to qualitative labeling, the sentiment intensity needs to be annotated quantitatively. A lot of the existing lexicons are manually annotated, including WordNet [8], General Inquirer [9], and HowNet [10]. To avoid the low efficiency and subjectivity of manual work, bootstrapping methods have been widely used. It is usually assumed that several seed words of known polarities are provided and different heuristics are adopted as the propagation strategy to infer the unknown sentiment polarities of other words. He sent the entries of HowNet into Google search and selected seed words according to the count of search results [11]. Li et al. introduced Pagerank to determine the polarity of words [12]. Each word is taken as a node in a graph, and HowNet is used to calculate the semantic similarity between seed words and candidate words as edge weights. The performance of these supervised methods is dependent on or limited by the accuracy of the third party tools or data. One possible way to solve this problem is to use unsupervised methods to obtain sentiment intensity from other corpuses or semantic resources. Colace et al. [13] construct the Mixed Graph of Terms by extracting 2-tuples like ⟨side, evaluation⟩ from comment text, and each item's intensity is inferred according to the domain knowledge in the Mixed Graph. Mukkamala et al. [14] define the expression of emotion as a fuzzy set of 4 elements ⟨topic, keyword, object, and tendency⟩. The relation strength of every 2 sets is determined through a membership function based on set theory and fuzzy logics. Turney and Littman [15] propose a semantic classification method based on emotional phrases. He first extracts the adjective or adverb phrases according to several defined templates and calculates the mutual information between words and phrases to determine the tendency and intensity of sentiment words. These unsupervised methods offered a lot of experience and help to us, but there is still much dependence on the accuracy of choosing, recognizing, and extracting various kinds of relationships of sentiment items.

Therefore, it is important to optimize the collection of sentiment lexicon entries and the intensity labeling. Chen et al. [16] construct polar square error function to decide if two entries have the same sentiment tendency and present an iterative expansion method. Turney and Littman [15] try to rationalize the intensity assignment by comparing the cooccurrence parameter of entries and seed words. Wang et al. [17] and Jo and Oh [18] both take tendency annotation as a by-product of sentiment classification task and yet failed to evaluate the annotation's quality. Some scholars try to introduce synonymous or antonymous relationships into the evaluation framework to optimize the intensity labeling [19, 20]. Compared with our work, the optimization framework of mentioned works is relatively simple and fails to take multisentiment words into consideration, which may express different tendency in various contexts.

Considering the above points, this paper presents an unsupervised model of automatic construction of a multi-sentiment lexicon based on WLI neural network language model [21] and a global optimization framework. The main contributions of this paper are as follows:

(1) We propose a new categorization of human emotions, which makes the linguistic features more suitable for computational analysis.

(2) We define the converting constraint set of distance and sentiment intensity and present an automatic construction model based on WLI language model.

(3) We present a global optimization framework based on several manually annotated semantic resources, to improve the semantic description of our lexicon SentiRuc.

3. Automatic Construction of SentiRuc

In this section, we present the "5 pairs with 10 polarities" categorization of human emotions and automatically annotate the multisentiment lexicon SentiRuc by defining the converting constraint set of distance and sentiment intensity. We also investigate the emotional disambiguation of multiple affective words in this section.

We integrated the entries of NTUSD dictionary, HowNet lexicon, and the DUT Ontology as the entries of our SentiRuc lexicon, which contains a total of 14250 emotional words.

3.1. WLI Language Model and the Categorization of Human Emotions. Traditional binary sentiment labeling has gradually become unable to meet the development of sentiment analysis tasks. The primary work of multiple sentiment labeling is the categorization of human emotions. Section 2 has discussed relevant achievements and existing problems. This paper takes the achievement of psychology, linguistics theory, and computation characteristics into consideration and categorizes human emotions to 10 categories: happy-sad, like-hate, believable-unexpected, gratitude-angry, and complementary-critical. Each pair contains 2 opposite sentiment polarities. Our goal is to annotate each sentiment word W with a 10-dimensional sentiment vector Senti(W), and the value of each dimension represents the intensity of the corresponding sentiment tendency. In the following research, the 10 words are directly adopted as seed words.

Words contain very rich meanings, and statistical language models are used to extract those semantic features. Given a corpus, neural network language model could map words into a high dimensional continuous space. Word2Vec is a tool based on deep learning and is released by Google in 2013, which adopts two main language models: the continuous bag of words model and continuous skip-gram model [22]. Mikolov et al. also found that the representations have very good linear semantic characteristics [23], so, in 2015, WLI neural network language model is presented to decrease the model complexity [21]. We accumulate the offset of corresponding dimensions of two-word representations as the linear semantic distance between them and further investigate how coordinate offset could affect word similarity. In this paper, we use Sogou News Corpus as the training set, which contains about 1.1 million different words.

3.2. Converting the Distances of Words into Word Similarities. All word representations are located in a high dimensional vector space, in which we determine an entry's polarity and intensity by computing the distance between the entry and seed words. However, there are many words that could express, for example, happiness. And it is difficult to choose one as the only seed of "happy." Here, to decrease the deviation caused by subjectivity, we use coordinate offset of word representations to list the 50 nearest neighbors of "happy" and then manually choose several words as the seed set of "happy." For example, we collect all distances between "bittersweet" and "happy" seeds and take the average distance as the distance between "bittersweet" and "happy" emotion. For any word W, we can obtain a 10-dimensional distance vector Dis(W) and each dimension of Dis(W), respectively, represents the distance between W and happy, like, believable, gratitude, complimentary, sad, hate, unexpected, angry, and critical

$$\begin{aligned}\text{Dis (bittersweet)} = (&1.13, 3.09, 3.70, 2.08, 4.45, 1.34,\\ &4.14, 2.56, 5.41, 2.72)\,.\end{aligned} \tag{1}$$

Previous research pointed out that, generally, a word mainly contains only one or two emotions [5], so we preserve the minimum one or two distances in Dis(W) as effective distances. Larger distances will be abandoned, which means that those sentiments with lower similarities will be eliminated. If threshold T is assigned with 3.00 for the word "bittersweet," only 2 distances in Dis(W), happy 1.13 and sad 1.34, will be retained, because the sum of the 2 distances has not reached T. That could be interpreted as "happy" and "sad" being the main sentiments contained in "bittersweet." Only these two distances are retained and used in the followup work and only these 2 distances are retained and used as "effective distances" in the followup work.

Paper [23] points out that linear coordinate offset between word representations is directly associated with words' semantic similarities. Therefore, we could annotate a word W's polarity intensity according to the coordinate offset between W and seed words in the vector space of word representations. Considering there could be more than one effective distances in Dis(W), it is necessary to investigate how different distributions of those distances impact words' similarities. To solve the problem of converting distance vector Dis(W) to sentiment vector Senti(W), we define 3 converting constraints.

Constraint 1 (diversity constraint). Each dimension of Senti(W) is denoted as Senti(W)[i] (i is an integer ranging from 1 to 10), indicating word W's intensity of each sentiment category. Senti(W)[i] shall be negatively correlated with the count of effective distances Count(Dis(W)), because it is observed that words with more effective distances usually lie farther away from each sentiment category, which could be interpreted as the sentiment intensities being "distracted" by various polarities. For example, "rage" is only 1.92 away from the "angry" category, while "unfair" is 3.38 away from "angry" and 5.05 away from "critical."

Constraint 2 (self constraint). Each dimension of Dis(W) is denoted as Dis(W)[i] (i is an integer ranging from 1 to 10), indicating the distance between word W and each sentiment category. The intensity of a certain sentiment Senti(W)[i] shall be negatively correlated with the corresponding distance Dis(W)[i]. The fact is that, in word representations, smaller distance indicates more semantic or pragmatic similarities.

Constraint 3 (global contrast constraint). The intensity of a certain sentiment Senti(W)[i] shall be negatively correlated with the ratio of Dis(W)[i] and the average effective distance Avg(Dis(W)). In a language, human habits cause big difference in word frequencies, and collocation of words also

divides words into various clusters. These both impact the quantized word representations. For example, the effective distance vector of "enjoy" is (2.09, 1.11, 0, 0, 0, 0, 0, 0, 0, 0) and that of "enchanted" is (5.26, 3.87, 0, 0, 0, 0, 0, 0, 0, 0). The global contrast constraint is used to eliminate this disparity.

From the converting constraints set we could derive the generating formula of W's sentiment vector Senti(W) as follows:

$$\text{Senti}\,(\text{W})\,[i] = \text{Diverge} \cdot \text{Self} \cdot \text{Contrast}. \tag{2}$$

Senti(W)[i] indicates word W's intensity of each sentiment category. The formula contains three factors: the factor of diversity constraint Diverge, the factor of self constraint Self, and the factor of global contrast constraint Contrast. These factors can be, respectively, expressed as follows:

$$\text{Diverge}\,(\text{Dis}\,(\text{W})) = \frac{C_0}{\alpha \cdot \text{Count}\,(\text{Dis}\,(\text{W})) + C_1}, \tag{3}$$

$$\text{Self}\,(\text{Dis}\,(\text{W})) = \left(\frac{C_2}{\text{Dis}\,(\text{W})\,[i] + C_2}\right)^{\beta}, \tag{4}$$

$$\text{Contrast}\,(\text{Dis}\,(\text{W})) = \left(\frac{\text{Avg}\,(\text{Dis}\,(\text{W}))}{\text{Dis}\,(\text{W})\,[i]}\right)^{\gamma}. \tag{5}$$

In formulas (3), (4), and (5), Count(Dis(W)) represents the count of effective distances and Avg(Dis(W)) is the average value of effective distances. According to constraints 1, 2, and 3, the positive or negative correlation has already been illustrated by denominators or numerators in formulas (3), (4), and (5). C_0, C_1, and C_2 are constants. In the experiments in Section 5, we will introduce the assignments of C_0, C_1, and C_2. The 3 parameters α, β, and γ,respectively determine the effect of each constraint. The optimal parameters can be trained through the optimization framework (Section 4).

Finally, to every sentiment word W, we annotate it in our sentiment lexicon with a 10-dimensional vector Senti(W). The value in each dimension represents the similarity between W and this sentiment, that is, W's intensity of this sentiment.

3.3. Sentiment Tendency Disambiguation Based on Word Distribution Density. In Section 3 we introduced an automatic method to identify a word's polarity and intensity. But some words convey different sentiment polarities in different contexts. It would be inappropriate to annotate such words in only one sentiment vector, so we investigate sentiment disambiguation in this section. Chen et al. [24] pointed out that "sentiment disambiguation is different from word sense disambiguation" because, in a general sentiment lexicon, a word's sentiment tendency is not directly correlated with its meaning.

We use a hybrid approach in screening multisentiment words from our lexicon's vocabulary. So far there is no effective method for automatic selection of multisentiment words. We attempted to extract words which appear in different synonym sets in "HIT Tongyicicilin" and "Synonym Lexicon for Pupils" and take these words as the candidate set S of multisentiment words. However, S shows good precision but poor recall. For example, "naive" could convey both positive and negative senses but is not covered in the candidate set. We finally decide to manually select multisentiment words from "HIT Tongyicicilin" and "Synonym Lexicon for Pupils" and take these words as multisentiment word set $\text{S}_{\text{MultiSenti}}$ which include altogether 148 entries.

Then, 113694 sentences containing words in $\text{S}_{\text{MultiSenti}}$ are selected from Sogou News Corpus, and the sentiment tendency of these words is annotated with a positive or negative label. In a context window size of 16, the distribution density of each context word is extracted and used as a feature of SVM classifier. The distribution density of a context word CW can be obtained by

$$\begin{aligned} \rho_{\text{CWpositive}} &= \frac{\text{Count}\,(\text{CWpositive})}{\text{Count}\,(\text{CW})}, \\ \rho_{\text{CWnegative}} &= \frac{\text{Count}\,(\text{CWnegative})}{\text{Count}\,(\text{CW})}. \end{aligned} \tag{6}$$

Count(CWpositive) represents the count of context word CW in all sentences which have a positive W. Count(CWnegative) is the count of CW in all sentences that have a negative W. Count(CW) is the total count of CW in all sentences where W appears.

After the tendency disambiguation, a multisentiment word W is split and segmented as two independent cases $\text{W}_{\text{positive}}$ and $\text{W}_{\text{negative}}$. The word representations would then be trained again, and the sentiment vectors of $\text{W}_{\text{positive}}$ and $\text{W}_{\text{negative}}$ could be generated through formula (2).

4. Global Optimization Framework

Section 3 presented a converting constraint set, and our lexicon SentiRuc is preliminarily generated. This section establishes a unified form of evaluation function to study the effects of various constraints. We've collected data from HIT Tongyicicilin, Synonym Lexicon for Pupils, Antonym Lexicon for Pupils, and the dataset of NLPCC 2013 Competition and NLPCC 2014 Competition. These datasets are all manually constructed resources and thus could be regarded as gold standards. The error function is used to evaluate the deviation between SentiRuc and those gold standards, and our goal is to find a set of parameters that minimizes the deviation.

4.1. Synonymous Relationship. If W_1 and W_2 are annotated as a pair of synonyms in HIT Tongyicicilin or Synonym Lexicon for Pupils, we can infer that their sentiment polarities and intensities tend to be similar. To formalize this intuition, we accumulate the deviation of sentiment intensity of W_1 and W_2 on corresponding dimensions in SentiRuc. The error function is shown as follows:

$$f_1\,(D) = \frac{1}{|D_1|}\sum_{D_1,i}\frac{\text{Senti}\,(\text{W}_1)\,[i] - \text{Senti}\,(\text{W}_2)\,[i]}{\text{Senti}\,(\text{W}_1)\,[i] + \text{Senti}\,(\text{W}_2)\,[i]}. \tag{7}$$

Senti(W)[i] indicates word W's intensity of each sentiment category. In formula (7), W_1 and W_2 are labeled

as synonyms and Pair(W_1, W_2) $\in D_1$, $|D_1|$ is the count of synonym pairs. $f_1(D)$ represents the average deviation of sentiment intensity of W_1 and W_2 on corresponding dimensions, when W_1 and W_2 are in both SentiRuc and synonym resources. $f_1(D)$ varies with parameters α, β and γ in Formulas (3), (4), and (5).

4.2. Antonymous Relationship. If W_1 and W_2 are annotated as a pair of antonyms in Antonym Lexicon for Pupils, we can infer that they shall have opposite sentiment polarities, where the intensities tend to be similar. In accordance with this intuition, we accumulate the deviation of sentiment intensity of W_1 and W_2 on opposite dimensions in SentiRuc. The error function is shown as formula (8)

$$f_2(D) = \frac{1}{|D_2|} \sum_{D_2,i,j} \frac{\text{Senti}(W_1)[i] - \text{Senti}(W_2)[j]}{\text{Senti}(W_1)[i] + \text{Senti}(W_2)[j]}. \quad (8)$$

Senti(W)[i] indicates word W's intensity of each sentiment category. In formula (8), W_1 and W_2 are labeled as antonyms (Pair(W_1, W_2) $\in D_2$), and i, j are the indices of opposite sentiments in sentiment vectors of W_1 and W_2. $|D_2|$ is the count of antonym pairs. $f_2(D)$ represents the average deviation of sentiment intensity of W_1 and W_2 on opposite dimensions, when W_1 and W_2 are in both SentiRuc and antonym resources. $f_2(D)$ varies with parameters α, β, and γ in formulas (3), (4), and (5).

4.3. Sentiment Ratings at the Sentence Level. If the annotation of SentiRuc could contribute more in relevant tasks, the sentiment classification result using SentiRuc shall be closer to human judgment than that using other lexicons. We select 6000 sentences from the dataset of NLPCC 2013 Competition and NLPCC 2014 Competition and label the sentences with a "main sentiment" and an optional "subsentiment," which are both involved in the 10 sentiment categories of SentiRuc. For a certain set of parameters in formulas (3), (4), and (5), we generate an individual annotation of SentiRuc for the sentiment classification task. The sentence-level error function is constructed by Jaccard similarity of classification result and labeled result, represented as follows:

$$f_3(D) = \frac{1}{|D_3|} \sum_{D_3} \text{Jaccard}(\text{Label}(d_i), \text{Sentence}(d_i)). \quad (9)$$

d_i is the identifier of each sample. $|D_3|$ is the number of sentences contained in the dataset. Label(d_i) represents the labeled sentiment vector of a sentence and Sentence(d_i) is the classified sentiment vector using SentiRuc. $f_3(D)$ shows the average of Jaccard similarity of each sentence's labeled result and classification result. $f_3(D)$ varies with parameters α, β, and γ.

4.4. The Global Error Function. By combining the above three evaluation methods we obtain the global optimization framework based on manually constructed resources, as shown in Figure 1.

The global error function is

$$f(D) = f_1(D) + f_2(D) + f_3(D). \quad (10)$$

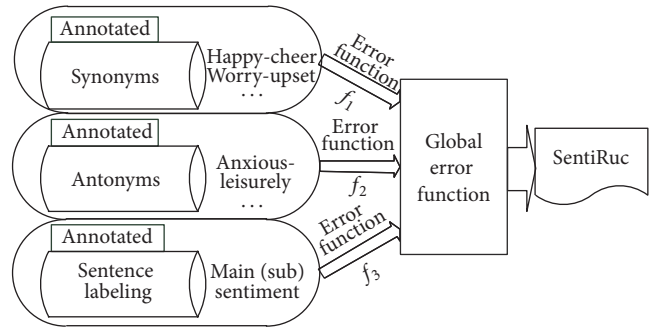


Figure 1: The global optimization framework.

The global error $f(D)$ varies with parameters α, β, and γ in formulas (3), (4), and (5). By minimizing $f(D)$ we can find the optimal parameter set argmin($f(D)$).

5. Experiments

We first evaluate the generating process of SentiRuc and then verify the availability of SentiRuc. To evaluate the rationality of the generating process, we design the parameters tuning experiment to prove the rationality of constraint set (Section 5.1) and verify the validity of sentiment tendency disambiguation method (Section 5.2). To test the availability of SentiRuc, we compare the qualitative and quantitative annotation of SentiRuc with other lexicons (Section 5.3) and investigate the performance of different lexicons in sentiment classification tasks (Section 5.4). NTUSD Lexicon of Taiwan University, HowNet sentiment lexicon, and DUT Ontology are all involved in the experiments.

In all experiments, the threshold distance value T is assigned with Avg(Dis(W)) so that only one or two kinds of sentiments would remain for each word. The parameter C_0 in formula (3) is set to 10, representing the number of sentiment categories. C_1 is set to 8, which means the number of sentiment categories minus the maximum remaining sentiments ($10 - 2 = 8$). C_2 is set to 3.38, representing the average coordinate offset between every two words included in SentiRuc. The dimension number of word representations is 60. Either more or less dimensions would increase the value of error function of the intensity annotating result.

Sogou News Corpus (3.17 GB) is used as the training text set. After segmentation by ICTCLAS 5.0 developed by Chinese Academy of Sciences, this corpus contains about 0.83 billion words, and the vocabulary size is 1,104,914. We do not have any other preprocessing of the data, so it can be ensured that every n-gram sample is a real Chinese word sequence and also that the word representations can show the actual semantic distribution of each word.

5.1. Evaluation of the Generating Constraint Set. Section 3 introduced how to automatically annotate the multisentiment lexicon SentiRuc by defining the converting constraint set of distance and sentiment intensity. Section 4 presented a global optimization framework to optimize the parameters. We first set α, β, and γ to 1 as the baseline experiment. If a parameter

Table 1: The tuning of generating parameters.

	α	β	γ	$f(D)$
Baseline	1	1	1	2.301
Dropping one constraint	0	1	1	2.402
	1	0	1	2.660
	1	1	0	2.599
Parameter tuning	1.875	1	1	2.123
	1.875	1.075	1	2.046
	1.875	1.075	1.145	1.972

is set to zero, it can be regarded as if this parameter is ignored. We conduct some contrast experiments where one parameter is dropped out in each experiment. The experimental result of $f(D)$ with the parameters is shown in Table 1.

It can be seen that dropping any constraint would increase the global error, which indicates that all constraints are useful in computing the intensity of SentiRuc. When β is dropped, $f(D)$ increases the most, which suggests that the self constraint contributes the most. It means that the intensity of a certain sentiment is remarkably negatively correlated with the corresponding distance, which proves the rationality of our method based on word representations. Then we try to find the optimal parameter set through variable-controlling method. As shown in the bottom three rows, the global error is further decreased, and the optimal parameter set is listed in the bottom line.

5.2. Evaluation of Tendency Disambiguation. Section 3.3 introduced how the 148 multisentiment words are selected. From Sogou News Corpus we collect sentences which contain these multisentiment words and label a multisentiment word W with "1" when W expresses positive tendency and with "2" if W contains negative tendency. To ensure excellent label result, eight Chinese native speakers participated in the annotating work. Every researcher made independent annotation of about 50 thousand sentences and each sentence is annotated by 4 researchers. If there is confliction in a sentence's label result, we made the final result through a panel discussion. In total, 113,694 sentences are annotated with positive tag or negative tag.

According to the labeled result, the tendency disambiguation algorithm based on word distribution density, which is introduced in Section 3.3, is used in the experiment. The experimental results of tenfold cross validation are shown in Table 2.

The overall disambiguation accuracy of all 148 words in the 113,694 sentences reaches 95.52%. The entries "epigone" and "yes-man" get the lowest accuracy, mainly due to the limited training data caused by their low occurrence. Generally, this experiment result shows that our disambiguation algorithm can effectively distinguish different tendencies of a word.

5.3. Evaluation of the Annotation's Quality of Sentiruc. The sentiment polarity and intensity of SentiRuc are both drawn from a Chinese corpus of GB grade level; therefore, its semantic description should be closer to actual semantic distribution than manually constructed lexicons. We try to evaluate the annotation quality of several existing lexicons by analyzing their sentiment category consistency (qualitative evaluation) and sentiment intensity consistency (quantitative evaluation). Sentiment category consistency examines the similarity of synonyms' (or antonyms') tendency annotation in SentiRuc. Sentiment intensity consistency refers to the similarity of synonyms' (or antonyms') intensity annotation in SentiRuc.

Table 2: Experiments of sentiment disambiguation.

	Target	Samples	Accuracy
Overall	148 words	113694	95.52%
Highest accuracy	滋生 breed	3095	98.71%
	幼稚 naive	1924	98.70%
Lowest accuracy	息事宁人 yes-man	281	87.90%
	萧规曹随 epigone	130	61.54%

Table 3: Evaluation of synonyms in each lexicon.

Lexicons	Synonyms	Category consistency	Intensity consistency
NTUSD	2179	87.29%	—
HowNet	2500	89.04%	—
DUT	2500	88.44%	70.89%
SentiRuc	2500	91.88%	92.54 %

HIT Tongyicicilin and Synonym Lexicon for Pupils contain 55,265 synonyms, from which we selected 2500 synonyms as the test dataset $S_{syn2500}$. The 1774 antonyms in Antonym Lexicon for Pupils are taken as test dataset $S_{ant1774}$. Multisentiment words are not included in $S_{syn2500}$ or $S_{ant1774}$. The intensity consistency $\text{Value}_{synonym}$ and intensity consistency $\text{Value}_{antonym}$ can be expressed as

$$\begin{aligned}
&\text{Value}_{synonym} \\
&\quad = 1 - \frac{1}{|D_{same}|} \sum \frac{\text{Senti}(W_1)[i] - \text{Senti}(W_2)[i]}{\text{Senti}(W_1)[i] + \text{Senti}(W_2)[i]}, \\
&\text{Value}_{antonym} \\
&\quad = 1 \\
&\qquad - \frac{1}{|D_{oppo}|} \sum \frac{\text{Senti}(W_1)[i] - \text{Senti}(W_2)[i+5]}{\text{Senti}(W_1)[i] + \text{Senti}(W_2)[i+5]}.
\end{aligned} \tag{11}$$

Senti(W)[i] (i is an integer ranging from 1 to 5) indicates word W's intensity of each positive sentiment category. D_{same} and D_{oppo} represent the count of corresponding dimensions which are annotated with nonzero value.

The evaluation result of synonyms is in Table 3 and that of antonyms is shown in Table 4.

Tables 3 and 4 indicate that, in SentiRuc, the tendency annotation of synonyms and antonyms are both closer to manually labeled resources than other sentiment lexicons.

Table 4: Evaluation of antonyms in each lexicon.

Lexicons	Antonyms	Category consistency	Intensity consistency
NTUSD	1450	84.00%	—
HowNet	1772	86.51%	—
DUT	1774	85.40%	67.55%
SentiRuc	1774	87.94%	91.62 %

In the condition that each word's sentiment intensity is computed independently, the intensity consistency of synonyms and antonyms in SentiRuc reaches 92% and 91%. This score is up to 20 percentage points higher than the manual intensity annotation of DUT Ontology, which is far more than expected. The results prove the effectiveness of the converting constraint set and formula (2) and also indicate that SentiRuc has better semantic descriptiveness.

5.4. Evaluation of SentiRuc in Sentiment Analysis Tasks. This experiment investigates the performance of sentiment analysis task using different lexicons. 3,100 sentences are selected from NLPCC 2013 Competition and NLPCC 2014 Competition and 3,700 sentences containing one of the 148 multisentiment words are selected from Sina Microblog. All 6,800 sentences are labeled with a "main sentiment" and an optional "subsentiment" tag. We define 2-gram part of speech (2-POS) and 3-gram part of speech (3-POS) for every labeled sample and extract sentiment tendency features with the help of SentiRuc. SVM is used in the multivariate classification experiments. Compared with human annotation result, the accuracy of the multivariate classification reaches 62.0%.

In order to facilitate the comparison of different lexicons, we also conduct binary classification experiments (positive or negative). Each of the 6,800 sentences is labeled with a "positive" or "negative" tag by four Chinese native speakers. The other 3200 objective sentences without affection are also labeled with "neutral" and added in the test dataset. For each sentence, we extract the 2-POS and 3-POS features and identify sentiment features with the help of SentiRuc. We use SVM classifier to implement tenfold cross validation. In addition, we also investigate the performance of SentiRuc before and after the tendency disambiguation. The results can be evaluated by

$$\begin{aligned} \text{Precision} &= \frac{\text{Result\_Correct}}{\text{Result\_Proposed}} \times 100\%, \\ \text{Recall} &= \frac{\text{Result\_Correct}}{\text{Result\_Labeled}} \times 100\%, \\ F\text{-measure} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 100\%. \end{aligned} \tag{12}$$

Result_Correct is the number of sentences that are correctly labeled with "positive" (or "negative"). Result_Proposed is the number of sentences labeled with "positive" (or "negative") by SVM model. Result_Labeled is the number of sentences manually labeled with "positive" (or "negative"). The result is shown in Table 5.

Table 5: Sentiment classification based on different lexicons.

Result of positive text			
Lexicon	Precision	Recall	$F1$
NTUSD	0.603	0.375	0.462
HowNet	0.728	0.540	0.620
DUT	0.721	0.552	0.593
SentiRuc (before disambiguation)	0.744	0.588	0.657
SentiRuc (after disambiguation)	0.782	0.678	0.726
Result of negative text			
Lexicon	Precision	Recall	$F1$
NTUSD	0.480	0.319	0.383
HowNet	0.611	0.451	0.519
DUT	0.572	0.445	0.501
SentiRuc (before disambiguation)	0.633	0.468	0.538
SentiRuc (after disambiguation)	0.671	0.589	0.627

Table 5 indicates that the F-measure of positive and negative classification using SentiRuc is apparently higher than those using other lexicons. In all 6800 subjective sentences, the sentences containing multisentiment words account for 54.4% and such a high percentage results in an apparent difference before and after disambiguation. Such a high percentage also brings impact on the overall F-measure on general domain text, respectively, 0.726 and 0.627. Actually, on the 6300 sentences which do not contain any multisentiment word, the F-measure of positive text and negative text is, respectively, 0.817 and 0.742.

6. Conclusion

This paper presented an automatic construction and global optimization framework of a multisentiment lexicon SentiRuc. The main jobs include the categorization of human emotions, an automatic construction model based on WLI language model, a global optimization framework based on several manually annotated semantic resources, and the disambiguation of multisentiment words. The experiment in Section 5 indicates that SentiRuc performs well on general dataset. Particularly, in intensity labeling test, SentiRuc outperforms the second place by 21 percent, which proves that statistical language modeling performs outstandingly in the semantic representation of sentiments. Our lexicon is now available online (https://pan.baidu.com/s/1jHAInlG).

It is difficult to directly compare existing lexicons because of various sentiment categorizations. We will investigate appropriate evaluation method of multiclass sentiment classification tasks.

Although Section 5 has shown the outstanding performance of word representations in sentiment lexicon's construction, the unique features of word representations still bring problems to text mining tasks. Firstly, statistical

language models depend a lot on the correspondence of inner semantics and outer grammars; thus, it is of great significance to research how to comprehend and distinguish "similar" words, "related" words, and their association with word representations' generating models. Secondly, similar words' vectors only differ much at several specific dimensions and further research on this kind of characterization is needed. We will study weighted statistical language models and will investigate the feasibility and effect of introducing various vector operations into the estimation of semantic distances.

Competing Interests

The authors declare that there are no competing interests.

Acknowledgments

This research was supported by the National Science Foundation for Young Scientists of China under Grant 61601371, the National Natural Science Foundation of China under Grant 71271209, Beijing Municipal Natural Science Foundation under Grant 4132052, and Humanity and Social Science Youth Foundation of Ministry of Education of China under Grant 11YJC630268.

References

[1] A. Esuli and F. Sebastiani, "Sentiwordnet: a publiclyavailable lexical resource for opinion mining," in *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC '06)*, pp. 417–422, Genoa, Italy, 2006.

[2] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining," in *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC '10)*, pp. 2200–2204, May 2010.

[3] DataTang, "National Taiwan University: simplified Chinese emotional dictionary [EB/OL]," 2013, http://www.datatang.com/data/11837.

[4] L. Xu, H. Lin, and Y. Pan, "Constructing the affective lexicon ontology," *Journal of the China Society for Scientific and Technical Information*, vol. 27, pp. 180–185, 2008.

[5] Y. Liu, F. Chen, W. Kong et al., "Identifying web spam with the wisdom of the crowds," *ACM Transactions on the Web*, vol. 6, no. 1, article 2, 30 pages, 2012.

[6] C. E. Osgood, "The nature and measurement of meaning," *Psychological Bulletin*, vol. 49, no. 3, pp. 197–237, 1952.

[7] C. Quan and F. Ren, "Construction of a blog emotion corpus for Chinese emotional expression analysis," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, vol. 3, pp. 1446–1454, Association for Computational Linguistics, Singapore, 2009.

[8] WordNet and C. Fellbaum, *WordNet: An Electronic Lexical Database*, Bradford Book, 1998.

[9] General Inquirer(GI)[EB/OL], 2012, http://www.wjh.harvard.edu/~inquirer/.

[10] Z. Dong, "Hownet sentiment lexicon [CP/OL]," 2012, http://www.keenage.com.

[11] F. He, "Orientation analysis for Chinese blog text based on semantic comprehension," *Journal of Computer Applicayions*, vol. 31, pp. 2130–2133, 2011.

[12] R.-J. Li, X.-J. Wang, and Y.-Q. Zhou, "Semantic orientation computing using PageRank model," *Journal of Beijing University of Posts and Telecommunications*, vol. 33, no. 5, pp. 141–144, 2010.

[13] F. Colace, M. de Santo, and L. Greco, "SAFE: a sentiment analysis framework for e-learning," *International Journal of Emerging Technologies in Learning*, vol. 9, no. 6, pp. 37–41, 2014.

[14] R. R. Mukkamala, A. Hussain, and R. Vatrapu, "Fuzzy-set based sentiment analysis of big social data," in *Proceedings of the 18th IEEE International Enterprise Distributed Object Computing Conference (EDOC '14)*, pp. 71–80, Ulm, Germany, September 2014.

[15] P. D. Turney and M. L. Littman, "Measuring praise and criticism: inference of semantic orientation from association," *ACM Transactions on Information Systems*, vol. 21, no. 4, pp. 315–346, 2003.

[16] L. Chen, W. Wang, M. Nagarajan, S. Wang, and A. P. Sheth, "Extracting diverse sentiment expressions with target-dependent polarity from twitter," in *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*, Kno.e.sis, Dublin, Ireland, 2012.

[17] H. Wang, Y. Lu, and C. Zhai, "Latent aspect rating analysis on review text data: a rating regression approach," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pp. 783–792, San Francisco, Calif, USA, July 2010.

[18] Y. Jo and A. Oh, "Aspect and sentiment unification model for online review analysis," in *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM '11)*, pp. 815–824, Hong Kong, February 2011.

[19] A. Neviarouskaya, H. Prendinger, and M. Ishizuka, "SentiFul: generating a reliable lexicon for sentiment analysis," in *Proceedings of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII '09)*, pp. 1–6, IEEE, Amsterdam, The Netherland, September 2009.

[20] S. Mohammad, C. Dunne, and B. Dorr, "Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*, pp. 599–608, August 2009.

[21] Z. Zhang, X. Yang, Q. Ma, and C. Xu, "Learning continuous word representations from large-scale corpus through linear approach," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '15)*, pp. 2678–2683, IEEE, Hong Kong, October 2015.

[22] T. Mikolov, *Statistical language models based on neural networks [Ph.D. thesis]*, Brno University of Technology, 2012.

[23] T. Mikolov, M. Karafiát, L. Burget et al., "Recurrent neural network based language model," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pp. 1045–1048, Chiba, Japan, September 2010.

[24] J. Chen, H. Lin, and Z. Yang, "Word emotion disambiguation based on Bayesian model," in *Proceedings of the 9th China National Conference on Computational Linguistics (CCL '07)*, 2007.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 9483646, 11 pages
http://dx.doi.org/10.1155/2016/9483646

Research Article

n-Gram-Based Text Compression

Vu H. Nguyen,[1] Hien T. Nguyen,[1] Hieu N. Duong,[2] and Vaclav Snasel[3]

[1]*Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam*
[2]*Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam*
[3]*Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava, Czech Republic*

Correspondence should be addressed to Hien T. Nguyen; hien@tdt.edu.vn

Received 21 May 2016; Revised 2 August 2016; Accepted 25 September 2016

Academic Editor: Geun S. Jo

Copyright © 2016 Vu H. Nguyen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose an efficient method for compressing Vietnamese text using *n*-gram dictionaries. It has a significant compression ratio in comparison with those of state-of-the-art methods on the same dataset. Given a text, first, the proposed method splits it into *n*-grams and then encodes them based on *n*-gram dictionaries. In the encoding phase, we use a sliding window with a size that ranges from bigram to five grams to obtain the best encoding stream. Each *n*-gram is encoded by two to four bytes accordingly based on its corresponding *n*-gram dictionary. We collected 2.5 GB text corpus from some Vietnamese news agencies to build *n*-gram dictionaries from unigram to five grams and achieve dictionaries with a size of 12 GB in total. In order to evaluate our method, we collected a testing set of 10 different text files with different sizes. The experimental results indicate that our method achieves compression ratio around 90% and outperforms state-of-the-art methods.

1. Introduction

According to [1], data compression is a process of converting an input data stream into another data stream that has a smaller size. A stream can be a file, a buffer in memory, or individual bits sent on a communications channel. The main objectives of data compression are to reduce the size of input stream and increase the transfer rate as well as save storage space. Typically, data compression techniques are classified into two classes, that is, lossless and lossy, based on the result of the decompression phase.

Text compression is a field of data compression, which uses the lossless compression technique to convert an input file to another form of data file. It cannot use the lossy compression technique because it needs to recover the exact original file from the compressed file. If lossy compression technique was used, the meaning of the decompression file will be different from the original file. Several techniques have been proposed for text compression in recent years. Most of them are based on the same principle of removing or reducing redundancies from the original input text file. The redundancy can appear at character, syllable, or word levels. This principle proposed a mechanism for text compression by assigning short codes to common parts, that is, characters, syllables, words, or sentences, and long codes to rare parts.

In recent years, several techniques have been developed for text compression. These techniques can be further classified into four major types, that is, substitution, statistical, dictionary, and context-based method. The substitution text compression techniques replace a certain longer repetition of characters with a shorter one. A technique that is a representative of these techniques is run-length encoding [2]. The statistical techniques usually calculate the probability of characters to generate the shortest average code length, such as Shannon-Fano coding [3, 4], Huffman coding [5], and arithmetic coding [6, 7]. The next type consists of dictionary techniques, which involve substitution of a substring of text by an index or a pointer code. They relate to a position in the dictionary of the substring. Representatives of these techniques are LZW [8], LZ77 [9], and LZ78 [10]. The last type is context-based techniques, which involve the use of minimal prior assumptions about the statistics of the text. Normally, they use the context of the text being encoded and the history of the text to provide more efficient compression. Representatives of this type are Prediction by Partial Matching (PPM) [11] and Burrow–Wheeler transform (BWT) [12]. Every

method has its own advantages and disadvantages when being applied in a specific field. None of the above methods has been able to achieve the best results in terms of compression ratio.

Normally, users will decide to choose the appropriate method based on their purposes. With systems that allow the reconstruction of information from the output not to be exactly the same as the input, we can use lossy methods, such as systems to compress images and voice messages. With systems that require the original data to be recovered exactly from the compressed data, we must use lossless methods, such as text compression systems.

This paper presents the first attempt at text compression using n-gram dictionaries, and the contribution has three attributes; that is, (1) it is a method for text compression using n-gram dictionaries, (2) it collects the text corpus of the Vietnamese language from the Internet and builds five n-gram dictionaries with nearly 500,000,000 n-grams, and (3) a test set of 10 different text files with different sizes to evaluate our new system and compare it with my two previous methods [13, 14] and also with other methods. The rest of this paper is organized as follows. Section 2 presents earlier work related to this effort. Section 3 presents our proposed method, and Section 4 presents our experiments and results. Our conclusions are presented in Section 5.

2. Related Work

In recent years, most text compression techniques have been based on dictionary, word, or character levels [15–18]. Reference [15] proposed a method to convert the characters in the source file to a binary code, where the most common characters in the file have the shortest binary codes and the least common have the longest. The binary codes are generated based on the estimated probability of the character within the file and are compressed using 8-bit character word length. In [16], the authors proposed a method that combined word with LZW. First, their method splits input text to word and nonword and then uses them as initial alphabet of LZW. Reference [17] proposed a technique to compress short text messages based on two phases. In the first phase, it converts the input text consisting of letters, numbers, spaces, and punctuation marks commonly used in English writing to a format which can be compressed in the second phase. In the second phase, it proposes a transformation which reduces the size of the message by a fixed fraction of its original size. In [18], the authors proposed a word-based compression variant based on the LZ77 algorithm and proposed and implemented various ways of sliding windows and various possibilities of output encoding. In a comparison with other word-based methods, their proposed method is the best. In these research, they do not consider the structure of words or morphemes in the text.

In addition, there are some approaches to text compression based on syllables, BWT. These approaches involve some languages that have morphology in the structure of words or morphemes (e.g., German, Arabic, Turkish, and Czech) such as in [19–23]. Reference [19] presented a new lossless text compression technique which utilizes syllable-based morphology of multisyllabic languages. The proposed method is designed to partition words into its syllables and then to produce their shorter bit representations for compression. The number of bits in coding syllables depends on the number of entries in the dictionary file. In [20], the authors proposed a genetic algorithm in syllable-based text compression. This algorithm was used to determine for the characteristic of syllables. These characteristics are stored into dictionary, which is part of the compression algorithm and it is not necessary to place them into compressed data. This leads to reduction of used space. In [21, 22], Lansky and his colleagues were the first to propose a method for syllable-based text compression techniques. In their papers, they focused on specification of syllables, methods for decomposition of words into syllables, and using syllable-based compression in combination of the principles of LZW and Huffman coding. In [23], the authors first proposed a method for small text file compression based on the Burrow–Wheeler transformation. This method combines the Burrow–Wheeler transform with the Boolean minimization at the same time.

In our previous papers for Vietnamese text compression [13, 14], we proposed a syllable-based method based on morphology and syllable dictionaries in [13]. With each morphosyllable, it is split into a consonant and a syllable, and they are compressed based on their corresponding dictionaries. This method has a compression ratio that converges to around 73%, and it is suitable for small text files. The method in [14] compressed text based on a trigram model; it splits a text file into trigrams and compresses these trigrams using a trigrams dictionary. This method achieves an encouraging compression ratio around 83%.

3. Proposed Method

In this section, we present a method for Vietnamese text compression using n-gram dictionaries. This model has two main modules. The first module is used for text compression and the second module performs decompression. Figure 1 describes our text compression model. In our model, we use n-gram dictionaries for both compression and decompression. We will describe the model in detail in the following subsections.

3.1. n-Gram Theory and Dictionaries

3.1.1. n-Gram Theory. In this paper, we use n-gram theory mentioned from [24]: an n-gram is a contiguous sequence of n items from a given sequence of a text or speech. An item can be a phoneme, a syllable, a letter, a word, or a morphosyllable. In general, an item is considered as an atomic unit. An n-gram of one item, two items, or three items is referred to as a "unigram," a "bigram," or a "trigram," respectively. Larger sizes are sometimes referred to by the number of items n, for example, "four-gram" and "five-gram."

3.1.2. Dictionaries. Since we focus on Vietnamese, we build five different Vietnamese dictionaries of unigram, bigram, trigram, four grams, and five grams corresponding to the

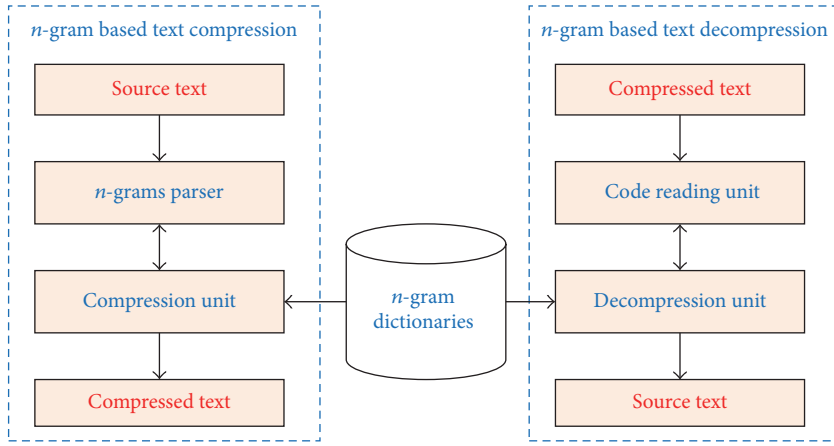


Figure 1: n-gram-based text compression model.

Table 1: n-gram dictionaries.

n-gram dictionary	Number of n-grams	Size (MB)
1	7,353	0.05
2	20,498,455	474
3	84,003,322	1,586
4	169,916,000	4,155
5	225,203,959	6,800

number of grams compressed. Table 1 shows these dictionaries with their number of n-grams and size. These dictionaries have been built based on a text corpus collected from the Internet. The size of the text corpus is around 2.5 GB. We use SRILM (http://www.speech.sri.com/projects/srilm/) to generate n-grams for these dictionaries. To increase the speed of searching in these dictionaries, we arranged them according to the alphabet. Table 1 describes the size and number of n-grams in each dictionary.

3.2. Compression. As presented in Figure 1, the compression module takes a source text as an input and then passes the text through two submodules, that is, n-grams parser and compression unit, to compress it. In following subsections, we explain in detail.

3.2.1. n-Gram Parser. n-gram parser has been used to read a source text file, splits it to sentences based on newline, and reads the number of grams in the combination with the result of the compression unit. In n-gram parser, we use five kinds of n-gram to store for unigram, bigram, trigram, four grams, and five grams. Based on the result of the compression unit, the n-gram parser decides how many grams will be read next. Algorithm 1 shows the pseudocode of this phase. If five grams was found in the five-gram dictionary, that is, index > 0, the force_four_gram_compression function would be called to encode all previous n-grams (unigram, bigram, trigram, and four grams); then the compress function would be called to encode this five grams. Next, the n-gram parser reads next five grams in the input string. Otherwise, it would split one leftmost gram of five grams for four grams and read one gram more from the input string for five grams. When the number of grams of four-gram was 4, it calls the four_gram_compression function.

```
input: The source text file
output: The encoded stream
(1) inputstring = read source text file
(2) count = number of grams in the inputstring
(3) while count ≥ 5 do
(4)     st5 = get first five grams of the inputstring
(5)     index = find(st5, five_gram_dict)
(6)     if index ≥ 0 then
(7)         force_four_gram_compression(st4)
(8)         outputstring += compress(index, 5)
(9)         delete first five grams of the inputstring
(10)        count −= 5
(11)    end
(12)    else
(13)        st4 += get first gram of the inputstring
(14)        delete first gram of the inputstring
(15)        count −= 1
(16)        if number of grams of st4 = 4 then
(17)            four_gram_compression(st4)
(18)        end
(19)    end
(20) end
(21) if count > 0 then
(22)    four_gram_compression(inputstring)
(23) end
```

Algorithm 1: Pseudocode of the compression phase.

Algorithm 2 shows the pseudocode of the four_gram_compression function. This function is used to compress four grams if it occurs in four-gram dictionary. Otherwise, it splits one leftmost gram of the four-gram variable for the trigram variable. Similar to this function, we have the trigram_compression, the bigram_compression, and the unigram_compression function.

```
input: The four-gram string, in this case is st4
output: The encoded stream
(1) index = find(st4, four_gram_dict)
(2) if index ≥ 0 then
(3)     force_trigram_compression(st3)
(4)     outputstring += compress(index, 4)
(5)     delete content of st4
(6) end
(7) else
(8)     st3 += first gram of st4
(9)     delete first gram of st4
(10)    if number of grams of st3 = 3 then
(11)        trigram_compression(st3)
(12)    end
(13) end
```

Algorithm 2: Pseudocode of the four_gram_compression.

```
input: The four-gram string, in this case is st4
output: The encoded stream
(1) while number of grams of st4 > 0 do
(2)     st3 += first gram of st4
(3)     delete first gram of st4
(4)     if number of grams of st3 = 3 then
(5)         trigram_compression(st3)
(6)     end
(7) end
(8) force_trigram_compression(st3)
```

Algorithm 3: Pseudocode of the force_four_gram_compression.

The force_four_gram_compression is called to encode all four-gram, trigram, bigram, and unigram when five-gram variable is found in the five-gram dictionary. Similar to this function, we have the force_trigram_compression, the force_bigram_compression, and the force_unigram_compression function (Algorithm 3).

3.2.2. Compression Unit. The compression unit uses the result from the n-gram parser to decide how many grams will be compressed and what kind of n-gram dictionaries should be used. Based on the number of n-grams in each dictionary, we will construct the number of bytes to encode each n-gram corresponding to the dictionary. Table 2 describes the number of bytes used to encode each n-gram of each dictionary.

To classify the dictionary that was used to encode each n-gram and the other cases, we use three most significant bits (MSB) of the first byte of each encoded byte. Table 3 describes the value of these bits corresponding to each dictionary.

The index of each n-gram corresponding to each dictionary is encoded in the bits after the first three bits of the first byte. As seen in Table 3, there are two special cases for the n-gram dictionary: a newline and a unigram that does not appear in the unigram dictionary corresponding to a value of "newline" and "others." In these cases, the compression unit will encode as follows:

(i) When the result received from the n-gram parser is the newline, the compression unit will encode the value "110" for the first three bits of MSB, and the next five bits of this byte will have the value "00000."

(ii) When the result is the others, the three MSB of the first byte are "111" and the next five bits of this byte present the number of bytes which were used to encode this gram.

Table 2: Number of encoded bytes for each n-gram of each dictionary.

n-gram dictionary	Number of n-grams	Number of bytes
1	7,353	2
2	20,498,455	4
3	84,003,322	4
4	169,916,000	4
5	225,203,959	4

Table 3: Value of three MSB and number of bytes.

n-gram dictionary	Value of three MSB	Number of bytes is read more
1	0 0 1	1
2	0 1 0	3
3	0 1 1	3
4	1 0 0	3
5	1 0 1	3
Newline	1 1 0	0
Others	1 1 1	Value of five bits after three first bits of current byte

3.3. Decompression. As seen in Figure 1, the decompression module takes a compressed text as an input and then passes the text through two submodules, that is, code reading unit and decompression unit, to decompress it. We explain in detail in following subsections.

3.3.1. Code Reading Unit. First, this unit reads the compressed text from the compression phase. This result becomes the input sequence of the code reading unit. The code reading unit splits this input sequence byte to byte. Then, it reads the first byte of the input sequence and splits and analyzes the first three bits of this byte to classify the dictionary to which this n-gram belongs. Based on this result, this unit will read more bytes from the input sequence. Table 2 shows the number of bytes that the code reading unit reads after the first byte according to the classification of the dictionary. After reading these bytes, it transfers them to the decompression unit and repeats its work until the input sequence is null.

```
    input: The encoded stream
    output: The decoded stream
(1) inputstring ← encodedstream
(2) while length of inputstring > 0 do
(3)     firstbyte = read first byte from the inputstring
(4)     delete first byte of the the inputstring
(5)     dict = get value of three bits of firstbyte
(6)     if dict ≤ 5 then
(7)         number = getnumberbytereadmore(dict)
(8)         bytereadmore = read number byte more from the inputstring
(9)         delete number byte of the inputstring
(10)        indexstring = get last five bits of the firstbyte + the bytereadmore
(11)        indexvalue = get value of the the indexstring
(12)        output += decompress(indexvalue, dict)
(13)    end
(14)    else if dict = 6 then
(15)        output += newline
(16)    end
(17)    else
(18)        number = value of five last bits of the firstbyte
(19)        bytereadmore = read number byte more from the inputstring
(20)        output += decode for the bytereadmore
(21)    end
(22) end
```

Algorithm 4: Pseudocode of the decompression phase.

3.3.2. Decompression Unit. This unit receives the results from the code reading unit. It decodes these results according to the classification of the dictionary as follows.

(i) Decode n-grams occurring in dictionaries

(1) Identifying the dictionary: based on the classification dictionary from the code reading unit
(2) Identifying the index of an n-gram in the dictionary: based on the value calculated from bytes that were read by the code reading unit
(3) Decode n-gram: when the classification of the dictionary has a value from one to five, the decompression unit decodes the n-gram in the dictionary based on the index of the n-gram

(ii) Decode n-grams that do not occur in dictionaries

(1) Decode newline: when the classification of dictionary is a "newline," it means that the value of the first three bits is 110. The decompression unit decodes a newline for this n-gram
(2) Decode others: when the classification of the dictionary is "others," based on the value of the remaining bits of the first byte, the decompression unit will decode all bytes after the first byte

After finishing the decoding for one n-gram or other cases, the decompression unit reads the next result from the code reading unit and repeats the decompression tasks to decode other n-grams or other cases until it reads the last byte. Algorithm 4 shows the pseudocode of the decompression phase.

3.4. Compression Ratio. Compression ratio is used to measure the efficiency of the compression method. The stronger the compression ratio is, the better the quality of this method is. The compression ratio can be calculated by

$$\mathrm{CR} = \left(1 - \frac{\text{compressed\_file\_size}}{\text{original\_file\_size}}\right) \times 100, \tag{1}$$

where original_file_size is size of the original file and compressed_file_size is size of the compressed file.

3.5. The Complexity of Our Method. Let n be the number of n-grams in the source text and a, b, c, d, and e be the number of five grams, four grams, trigrams, bigrams, and unigrams, respectively, in dictionaries. Let k be $\log_2(a) + \log_2(b) + \log_2(c) + \log_2(d) + \log_2(e)$. According to pseudocode from Algorithm 1, in the worst case, all five grams, four grams, trigrams, and bigrams do not occur in five grams, four grams, trigram, and bigram dictionary, respectively. Hence, the complexity of our method is $O(k * n)$.

3.6. Example

3.6.1. Compression Phase. Let us encode the following sequence using the n-gram approach.

Nén dữ liệu nhằm giảm kích thước dữ liệu để tăng tốc độ truyền cũng như tiết kiệm không gian lưu trữ

Assume that we have five dictionaries for unigram, bigram, trigram, four grams, and five grams, as seen in Table 4.

The n-gram parser first encounters the first five-gram *Nén dữ liệu nhằm giảm* and copies it to the five-gram variable. This pattern is not in the five-gram dictionary, so it splits the first gram of this pattern for the four-gram variable and concatenates the next gram of the input sequence to the five-gram variable. The content of the five-gram and four-gram variables becomes *dữ liệu nhằm giảm kích* and *Nén*,

Table 4: Five dictionaries.

(a) Unigram dictionary

Index	Entry
1	nhằm
2	lưu
3	trữ

(b) Bigram dictionary

Index	Entry
1	cũng như

(c) Trigram dictionary

Index	Entry
1	Nén dữ liệu

(d) Four-gram dictionary

Index	Entry
1	tiết kiệm không gian

(e) Five-gram dictionary

Index	Entry
1	giảm kích thước dữ liệu
2	để tăng tốc độ truyền

respectively. Then, it checks the number of grams in the four-gram variable, which is one at this time. In this case, the value is less than four; it bypasses the four_gram_compression and turns back to the five-gram variable. Because this pattern is not in the five-gram dictionary, similar to the first case, it splits the first gram of this five-gram to the four-gram variable and concatenates the next gram of the input sequence to the five-gram variable. The content of the five-gram and four-gram variables shall become *liệu nhằm giảm kích thước* and *Nén dữ*, respectively. Then, it checks the number of grams in the four-gram variable, which is two now. This value is less than four, similar to the first case; it turns back to five-gram variable. It repeats these operations until the content of the five-gram variable is *nhằm giảm kích thước dữ* and the four-gram variable is *Nén dữ liệu*. This five-gram pattern is not in five-gram dictionary, so it splits the first gram of this pattern for the four-gram variable and concatenates the next gram of the input sequence to the five-gram variable. The content of the five-gram and four-gram variables shall become *giảm kích thước dữ liệu* and *Nén dữ liệu nhằm*, respectively. It checks the number of grams in the four-gram variable, which is four now. It calls the four_gram_compression as presented in Algorithm 2. The four_gram_compression searches the four-gram pattern in the four-gram dictionary, which is not found in the four-gram dictionary. It splits the first gram of this pattern into the trigram variable. The content of the four-gram and the trigram variable becomes *dữ liệu nhằm* and *Nén*, respectively. Then, it checks the number of grams in the trigram variable, which is one at this time. So, it bypasses the trigram_compression, exits the four_gram_compression, and turns back to five-gram variable in Algorithm 1. The first five steps as seen in Table 5 show the content of the five-gram, four-gram, and trigram variables throughout these steps.

At Step 6, first, the *n*-gram parser checks the value of the five-gram variable in the five-gram dictionary. This pattern is in the dictionary; therefore, it calls the compression unit to encode all bigram, trigram, and four grams. Then, it encodes the five-gram. When the compression unit is finished, the *n*-gram parser reads the next five grams from the input sequence. In Table 5, Steps 6.1 to 6.4 show all substeps of Step 6 and in Table 6, Steps 6.2 to 6.4 show the encoder output sequence.

As seen in Table 5, at Step 6.1, the *n*-gram parser splits the first gram of the four-gram variable for the trigram variable, and the content of the four-gram and trigram variable shall become *liệu nhằm* and *Nén dữ*, respectively. Then, it checks the number of grams in the trigram variable, which is two at this time. So, it bypasses the trigram_compression and moves to Step 6.2. At Step 6.2, it continues splitting the first gram of the four-gram variable for the trigram variable. The content of the four-gram and trigram variables shall become *nhằm* and *Nén dữ liệu*, respectively. Next, it checks the number of grams in the trigram variable, which is three at this time. It then searches for this trigram in the trigram dictionary. Because this trigram is in the trigram dictionary, it calls the compression unit to encode bigram in the bigram variable. In this case, the bigram variable is null. It calls the compression unit to encode the trigram in the trigram variable and moves to the next substep. The encoded sequence of this trigram is shown in Table 6 at Step 6.2. The first three bits of this encoded sequence which have value *011* refer to trigram dictionary as seen in Table 3 and all remaining bits refer to the index of this trigram in the trigram dictionary.

At Step 6.3, the bigram and trigram variables are null; it counts the number of grams in the four-gram variable, which is 1 in this case; then it copies this gram to the unigram and searches for this unigram in the unigram dictionary. This unigram is in dictionary so it calls the compression unit to encode this unigram. The encoder output sequence of this unigram is shown in Table 6 at Step 6.3. At Step 6.4, it calls the compression unit to encode the five-gram in the five-gram variable, and the encoder output sequence of this five-gram is shown in Table 6 at Step 6.4. Then it reads the next five-gram in the input sequence to the five-gram variable. At this time, the content of the five-gram variable is *để tăng tốc độ truyền*.

The *n*-gram parser and the compression unit will process similar to previous cases for all remaining grams of the input sequence. The results of these steps are shown in Table 5 from Step 7 to Step 13.4. The encoder output sequences are shown in Table 6 from Step 7 to Step 13.4. The final encoder output sequence is the result of concatenation of all encoder output sequences from Step 6.1 to 13.4 in Table 6. The final encoder output sequence is

011000000000000000000000000000001|00100000
00000001|10100000000000000000000000000001
|10100000000000000000000000000010|01000000
00000000000000000000000001|10000000000000000
0000000000000001|00100000000000010|00100000
00000011.

Table 5: All steps and values of *n*-grams.

Step	Five-gram variable	Four-gram variable	Trigram variable	Bigram variable
1	Nén dữ liệu nhằm giảm			
2	dữ liệu nhằm giảm kích	Nén		
3	liệu nhằm giảm kích thước	Nén dữ		
4	nhằm giảm kích thước dữ	Nén dữ liệu		
5	giảm kích thước dữ liệu	dữ liệu nhằm	Nén	
6.1	giảm kích thước dữ liệu	liệu nhằm	Nén dữ	
6.2	giảm kích thước dữ liệu	nhằm		
6.3	giảm kích thước dữ liệu			
6.4	để tăng tốc độ truyền			
7	cũng như tiết kiệm không			
8	như tiết kiệm không gian	cũng		
9	tiết kiệm không gian lưu	cũng như		
10	kiệm không gian lưu trữ	cũng như tiết		
11	không gian lưu trữ	như tiết kiệm	cũng	
12	gian lưu trữ	tiết kiệm không	cũng như	
13.1	lưu trữ	tiết kiệm không gian		
13.2				lưu trữ
13.3				trữ
13.4				

Table 6: Encoder output sequences.

Step	Encoding of dictionary	Encoded sequence
6.2	011	00000000000000000000000000001
6.3	001	0000000000001
6.4	101	00000000000000000000000000001
7	101	00000000000000000000000000010
13.1	010	00000000000000000000000000001
13.2	100	00000000000000000000000000001
13.3	001	0000000000010
13.4	001	0000000000011

3.6.2. Decompression Phase. In this section, the encoder output sequence from the previous example is taken and is decoded using the decompression unit. The encoder output sequence in the previous example was

> *01100000000000000000000000000001|00100000 00000001|10100000000000000000000000000001 |10100000000000000000000000000010|01000000 00000000000000000000000001|10000000000000000 00000000000001|00100000000000010|00100000 00000011.*

The decompression unit uses the same dictionaries as the compression unit as seen in Table 4. It reads the first byte of the input sequence; the content of this first byte is *01100000*. The first three bits are split, and the value of these three bits is *011*. It finds the corresponding *n*-gram dictionary of these three bits and the number of bytes that is read more as presented in Table 3. In this case, the *n*-gram dictionary is the trigram dictionary and the number of bytes that is read more is 3. The decoder reads the next three bytes from the input sequence. The index of the entry was calculated based on the value of all remaining bits after the first three bits and the three bytes that is read more. The entry is determined based on this index. The decoder repeats these steps until it reads the last byte of the input sequence. Table 7 shows all steps and results of the decompression phase.

The final decoder output sequence is the result of concatenation of all decoder output sequences from Step 1 to Step 8 as presented in Table 7. With each decoder output sequence from Step 1 to Step 7, we add one space character before the concatenation. The final encoder output sequence is *Nén dữ liệu nhằm giảm kích thước dữ liệu để tăng tốc độ truyền cũng như tiết kiệm không gian lưu trữ*.

4. Experiments

We conducted an experiment to evaluate our method, using a dataset that is randomized collection from some Vietnamese news agencies. The dataset includes 10 files completely different in size and content.

In order to evaluate the effects of a combination of various *n*-gram dictionaries, we conducted three experiments with three kinds of systems. In the first case, we build a system with unigram, bigram, and trigram dictionaries. Next, we extend the first one with four-gram dictionary. Lastly, we extend the second one with five-gram dictionary. The results of the three experiments are shown in Table 8. As presented in Table 8, we find out that the compression ratio from the third case is

Table 7: All steps and the results of the decompression phase.

Step	First byte		Dict.	nbm bits to calculate index	Index value	Decoder output sequence
1	01100000	011	3	00000000000000000000000000001	1	Nén dữ liệu
2	00100000	001	1	0000000000001	1	nhằm
3	10100000	101	3	00000000000000000000000000001	1	giảm kích thước dữ liệu
4	10100000	101	3	00000000000000000000000000010	2	để tăng tốc độ truyền
5	01000000	010	3	00000000000000000000000000001	1	cũng như
6	10000000	100	3	00000000000000000000000000001	1	tiết kiệm không gian
7	00100000	001	1	0000000000010	2	lưu
8	00100000	001	1	0000000000011	3	trữ

Table 8: Compression ratio of three experience cases.

Number	OFS	CFS-C1	CR-C1	CFS-C2	CR-C2	CFS-C3	CR-C3
1	1,166	210	81.99%	166	85.76%	136	88.34%
2	2,240	362	83.84%	274	87.77%	222	90.09%
3	6,628	1,245	81.22%	999	84.93%	887	86.62%
4	12,224	1,954	84.02%	1,503	87.70%	1,179	90.36%
5	22,692	3,565	84.29%	2,652	88.31%	2,180	90.39%
6	49,428	7,638	84.55%	5,712	88.44%	4,538	90.82%
7	96,994	15,636	83.88%	12,359	87.26%	10,416	89.26%
8	156,516	24,974	84.04%	19,188	87.74%	15,889	89.85%
9	269,000	43,887	83.69%	34,182	87.29%	28,937	89.24%
10	489,530	80,685	83.52%	63,472	87.03%	54,117	88.95%

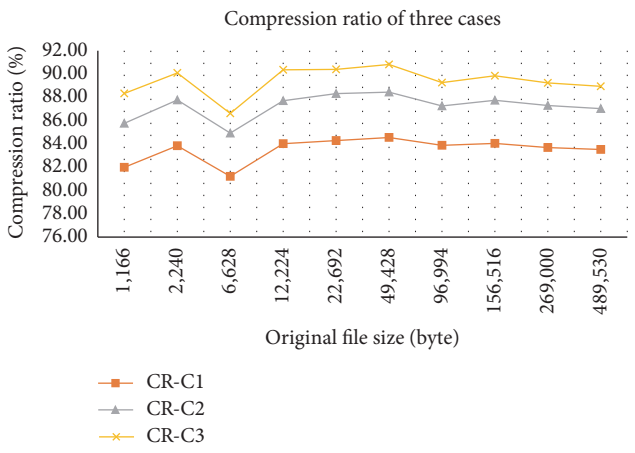


Figure 2: Comparison between the three cases.

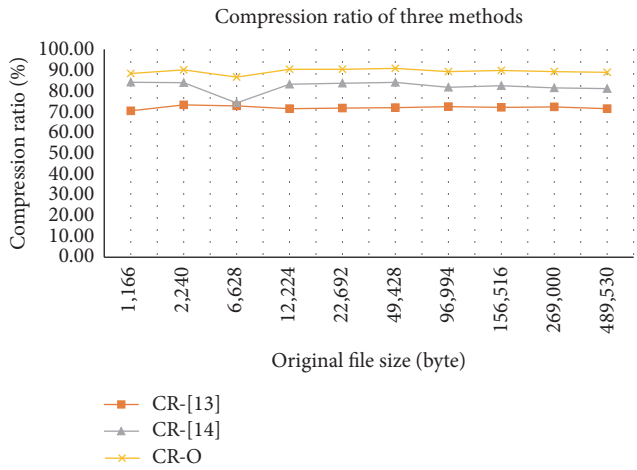


Figure 3: Compression ratio of our method [13, 14].

the best, follow-up is the second case, and the last one comes from the first case. The compression ratio in this section was used according to (1). In Tables 8, 9, and 10 and Figures 2, 3, 4, 5, and 6, we have some abbreviations and meanings as follows: OFS: original file size in byte; CFS: compressed file size in byte; CR: compression ratio; C1, C2, and C3: three cases above, respectively; O: our method; RAR: WinRAR; ZIP: WinZIP.

As seen in Figure 2, the compression ratio when we combine all five dictionaries is the highest.

In order to evaluate our method with the methods presented in [13, 14], we compress the input files using these methods. In Table 9, we show the results of the current method in 10 cases in comparison with the methods in [13, 14]. As shown in Table 9 and Figure 3, the compression ratio of our method is better than the methods presented in [13, 14] for any size of text in our test cases.

Table 10 and Figure 4 show the results of our method in comparison with those of other methods, such as WinZIP version 19.5 (http://www.winzip.com/win/en/index.htm), the

Table 9: CR of the current method with the methods presented in [13, 14].

Number	OFS	CFS-[13]	CR-[13]	CFS-[14]	CR-[14]	CFS-O	CR-O
1	1,166	345	70.41%	185	84.13%	136	88.34%
2	2,240	599	73.26%	359	83.97%	222	90.09%
3	6,628	1,803	72.80%	1,710	74.20%	887	86.62%
4	12,224	3,495	71.41%	2,057	83.17%	1,179	90.36%
5	22,692	6,418	71.72%	3,702	83.69%	2,180	90.39%
6	49,428	13,881	71.92%	7,870	84.08%	4,538	90.82%
7	96,994	26,772	72.40%	17,723	81.73%	10,416	89.26%
8	156,516	43,701	72.08%	27,434	82.47%	15,889	89.85%
9	269,000	74,504	72.30%	49,902	81.45%	28,937	89.24%
10	489,530	139,985	71.40%	92,739	81.06%	54,117	88.95%

Table 10: Compression ratio of our method, WinRAR, and WinZIP.

Number	OFS	CFS-O	CR-O	CFS-RAR	CR-RAR	CFS-ZIP	CR-ZIP
1	1,166	136	88.34%	617	47.08%	676	42.02%
2	2,240	222	90.09%	887	60.40%	946	57.77%
3	6,628	887	86.62%	2,052	69.04%	2,111	68.15%
4	12,224	1,179	90.36%	3,378	72.37%	3,442	71.84%
5	22,692	2,180	90.39%	6,162	72.85%	6,150	72.90%
6	49,428	4,538	90.82%	12,504	74.70%	12,286	75.14%
7	96,994	10,416	89.26%	21,389	77.95%	21,321	78.02%
8	156,516	15,889	89.85%	34,162	78.17%	34,362	78.05%
9	269,000	28,937	89.24%	56,152	79.13%	57,671	78.56%
10	489,530	54,117	88.95%	101,269	79.31%	108,175	77.90%

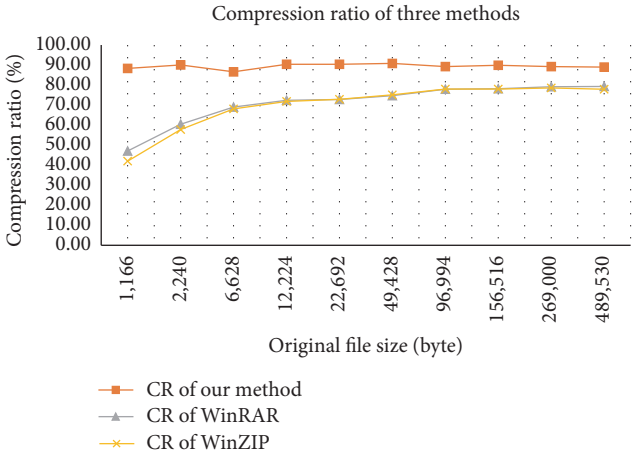


Figure 4: Compression ratio of our method, WinRAR, and WinZIP.

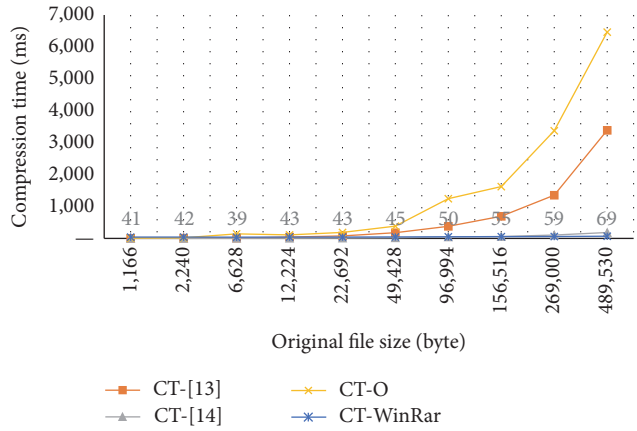


Figure 5: Compression time of four methods.

software combining LZ77 [8] and Huffman coding, and WinRAR version 5.21 (http://www.rarlab.com/download.htm), the software combining LZSS [25] and Prediction by Partial Matching [11]. The experimental results show that our method achieves the highest compression ratio on the same testing set.

Tables 11 and 12 and Figures 5 and 6 show the compression and decompression time of our method in comparison with the methods in [13, 14] and WinRAR, respectively. In these tables and figures, we have some abbreviations and meanings as follows: CT: compression time; DT: decompression time; RAR: WinRAR; O: our method; ms: millisecond.

As presented in Table 12 and Figure 5, the compression time of our method is higher than those of other methods.

Table 11: Compression time of four methods.

Number	File size	CT-[13]-ms	CT-[14]-ms	CT-O-ms	CT-RAR-ms
1	1,166	4	1	11	41
2	2,240	8	2	19	42
3	6,628	12	4	143	39
4	12,224	43	5	111	43
5	22,692	79	10	187	43
6	49,428	181	21	383	45
7	96,994	381	47	1,246	50
8	156,516	692	60	1,623	55
9	269,000	1,356	105	3,374	59
10	489,530	3,388	185	6,463	69

Table 12: Decompression time of four methods.

Number	File size	CT-[13]-ms	CT-[14]-ms	CT-O-ms	CT-RAR-ms
1	1,166	1	1	9	30
2	2,240	2	2	17	32
3	6,628	4	3	56	34
4	12,224	8	8	83	37
5	22,692	18	13	149	39
6	49,428	70	18	329	42
7	96,994	290	70	770	46
8	156,516	961	250	1,398	54
9	269,000	3,111	873	2,958	67
10	489,530	11,427	5,070	6,773	98

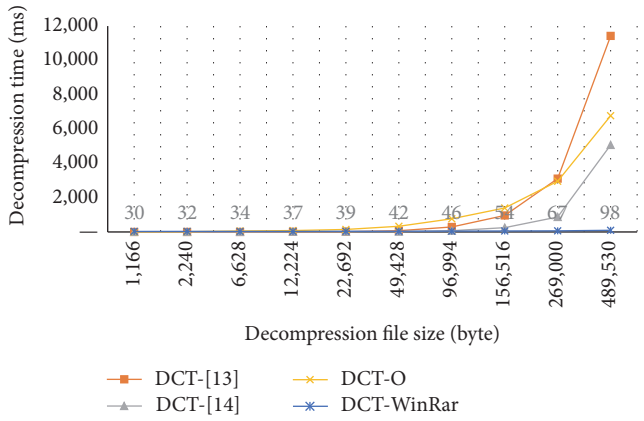


Figure 6: Decompression time of four methods.

As presented in Table 12 and Figure 6, the decompression time of our method is higher than [14] but it is slower than [13] and WinRAR.

5. Conclusions

In this paper, we present a novel method using n-gram dictionaries for text compression. We build five different n-gram dictionaries range from unigram to five grams from a 2.5 GB text corpus and obtain approximately 12 GB n-grams. We conduct experiments on a dataset of 10 files with different sizes and content in three different scenarios. The first scenario uses unigram, bigram, and trigram dictionaries. The second scenario extends the first one with four-gram dictionary and the final scenario extends the second one with five-gram dictionary. The experimental results show that our method achieves the performance comparable with those of state-of-the-art methods including WinZIP and WinRAR in terms of compression ratio, while it is slower than these two of WinZIP and WinRAR. Speeding-up looking-up process of dictionaries may lead to foster the running time of ours method. We put this perspective as a direction of research in future.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. Salomon and G. Motta, *Data Compression—The Complete Reference*, Springer, New York, NY, USA, 5th edition, 2010.

[2] A. H. Robinson and C. Cherry, "Results of a prototype television bandwidth compression scheme," *Proceedings of the IEEE*, vol. 55, no. 3, pp. 356–364, 1967.

[3] R. M. Fano, "The transmission of information," Tech. Rep., Massachusetts Institute of Technology, Research Laboratory of Electronics, Cambridge, Mass, USA, 1949.

[4] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[5] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[6] P. G. Howard and J. S. Vitter, "Arithmetic coding for data compression," *Proceedings of the IEEE*, vol. 82, no. 6, pp. 857–865, 1994.

[7] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[8] T. A. Welch, "Technique for high-performance data compression," *IEEE Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[9] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[10] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[11] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[12] M. Burrows and D. Wheeler, "A block-sorting lossless data compression algorithm," Digital SRC Research Report, 1994.

[13] V. H. Nguyen, H. T. Nguyen, H. N. Duong, and V. Snasel, "A syllable-based method for vietnamese text compression," in *Proceedings of the ACM 10th International Conference on Ubiquitous Information Management and Communication (IMCOM '16)*, p. 17, Danang, Viet Nam, January 2016.

[14] V. H. Nguyen, H. T. Nguyen, H. N. Duong, and V. Snasel, "Trigram-based vietnamese text compression," in *Recent Developments in Intelligent Information and Database Systems*, vol. 642 of *Studies in Computational Intelligence*, pp. 297–307, Springer, 2016.

[15] H. Al-Bahadili and S. M. Hussain, "An adaptive character wordlength algorithm for data compression," *Computers and Mathematics with Applications*, vol. 55, no. 6, pp. 1250–1256, 2008.

[16] J. Dvorskþ, J. Pokornþ, and J. Snásel, "Word-based compression methods and indexing for text retrieval systems," in *Proceedings of the 3rd East European Conference on Advances in Databases and Information Systems (ADBIS '99)*, pp. 75–84, Maribor, Slovenia, 1999.

[17] K. Kalajdzic, S. H. Ali, and A. Patel, "Rapid lossless compression of short text messages," *Computer Standards & Interfaces*, vol. 37, pp. 53–59, 2015.

[18] J. Platos and J. Dvorskþ, "Word-based text compression," CoRR abs/0804.3680, 2008.

[19] I. Akman, H. Bayindir, S. Ozleme, Z. Akin, and S. Misra, "A lossless text compression technique using syllable based morphology," *The International Arab Journal of Information Technology*, vol. 8, no. 1, pp. 66–74, 2011.

[20] T. Kuthan and J. Lansky, "Genetic algorithms in syllable-based text compression," in *Proceedings of the Dateso Annual International Workshop on Databases, Texts, Specifications and Objects*, Desna, Czech Republic, April 2007.

[21] J. Lansky and M. Zemlicka, "Text compression: syllables," in *Proceedings of the Dateso Annual International Workshop on Databases, Texts, Specifications and Objects*, pp. 32–45, Desna, Czech Republic, April 2005.

[22] J. Lansky and M. Zemlicka, "Compression of small text files using syllables," in *Proceedings of the Data Compression Conference*, Snowbird, Utah, USA, March 2006.

[23] J. Platoš, V. Snášel, and E. El-Qawasmeh, "Compression of small text files," *Advanced Engineering Informatics*, vol. 22, no. 3, pp. 410–417, 2008.

[24] P. Koehn, *Statistical Machine Translation*, Cambridge University Press, Cambridge, UK, 2009.

[25] J. A. Storer and T. G. Szymanski, "Data compression via textual substitution," *Journal of the ACM*, vol. 29, no. 4, pp. 928–951, 1982.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 3483528, 16 pages
http://dx.doi.org/10.1155/2016/3483528

Research Article

A New Data Representation Based on Training Data Characteristics to Extract Drug Name Entity in Medical Text

Mujiono Sadikin,[1,2] Mohamad Ivan Fanany,[2] and T. Basaruddin[2]

[1] *Faculty of Computer Science, Universitas Mercu Buana, l. Meruya Selatan No. 1, Kembangan, Jakarta Barat 11650, Indonesia*
[2] *Machine Learning and Computer Vision Laboratory, Faculty of Computer Science, Universitas Indonesia, Depok, West Java 16424, Indonesia*

Correspondence should be addressed to Mujiono Sadikin; mujiono.sadikin@mercubuana.ac.id

Received 27 May 2016; Revised 8 August 2016; Accepted 18 September 2016

Academic Editor: Trong H. Duong

Copyright © 2016 Mujiono Sadikin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

One essential task in information extraction from the medical corpus is drug name recognition. Compared with text sources come from other domains, the medical text mining poses more challenges, for example, more unstructured text, the fast growing of new terms addition, a wide range of name variation for the same drug, the lack of labeled dataset sources and external knowledge, and the multiple token representations for a single drug name. Although many approaches have been proposed to overwhelm the task, some problems remained with poor *F*-score performance (less than 0.75). This paper presents a new treatment in data representation techniques to overcome some of those challenges. We propose three data representation techniques based on the characteristics of word distribution and word similarities as a result of word embedding training. The first technique is evaluated with the standard NN model, that is, MLP. The second technique involves two deep network classifiers, that is, DBN and SAE. The third technique represents the sentence as a sequence that is evaluated with a recurrent NN model, that is, LSTM. In extracting the drug name entities, the third technique gives the best *F*-score performance compared to the state of the art, with its average *F*-score being 0.8645.

1. Introduction

The rapid growth of information technology provides rich text data resources in all areas, including the medical field. An abundant amount of medical text data can be used to obtain valuable information for the benefit of many purposes. The understanding of drug interactions, for example, is an important aspect of manufacturing new medicines or controlling drug distribution in the market. The process to produce a medicinal product is an expensive and complex task. In many recent cases, however, many drugs are withdrawn from the market when it was discovered that the interaction between the drugs is hazardous to health [1].

Information, or objects extraction, from an unstructured text document, is one of the most challenging studies in the text mining area. The difficulties of text information extraction keep increasing due to the increasing size of corpora, continuous growth of human's natural language, and the unstructured formatted data [2]. Among such valuable information are medical entities such as drug name, compound, and brand; disease names and their relations, such as drug-drug interaction and drug-compound relation. We need a suitable method to extract such information. To embed those abundant data resources, however, many problems have to be tackled, for example, large data size, unstructured format, choosing the right NLP, and the limitation of annotated datasets.

More specific and valuable information contained in medical text data is a drug entity (drug name). Drug name recognition is a primary task of medical text data extraction since the drug finding is the essential element in solving other information extraction problems [3, 4]. Among derivative work of drug name extractions are drug-drug interaction [5], drug adverse reaction [6], or other applications (information

retrieval, decision support system, drug development, or drug discovery) [7].

Compared to other NER (name entity recognition) tasks, such as PERSON, LOCATION, EVENT, or TIME, drug name entity recognition faces more challenges. First, the drug name entities are usually unstructured texts [8] where the number of new entities is quickly growing over time. Thus, it is hard to create a dictionary which always includes the entire lexicon and is up-to-date [9]. Second, the naming of the drug also widely varies. The abbreviation and acronym increase the difficulties in determining the concepts referred to by the terms. Third, many drug names contain a combination of nonword and word symbols [10]. Fourth, the other problem in drug name extraction is that a single drug name might be represented by multiple tokens [11]. Due to the complexity in extracting multiple tokens for drugs, some researchers such as [12] even ignore that case in the MedLine and DrugBank training with the reason that the multiple tokens drug is only 18% of all drug names. It is different with another domain; that is, entity names in the biomedical field are usually longer. Fifth, in some cases, the drug name is a combination of medical and general terms. Sixth, the lack of the labelled dataset is another problem; it has yet to be solved by extracting the drug name entities.

This paper presents three data representation techniques to extract drug name entities contained in the sentences of medical texts. For the first and the second techniques, we created an instance of the dataset as a tuple, which is formed from 5 vectors of words. In the first technique, the tuple was constructed from all sentences treated as a sequence, whereas in the second technique the tuple is made from each sentence treated as a sequence. The first and second techniques were evaluated with the standard MLP-NN model which is performed in the first experiment. In the second experiment, we use the second data representation technique which is also applied to the other NN model, that is, DBN and SAE. The third data representation, which assumes the text as sequential entities, was assessed with the recurrent NN model, LSTM. Those three data representation techniques are based on the word2vec value characteristics, that is, their cosine and the Euclidean distance between the vectors of words.

In the first and second techniques, we apply three different scenarios to select the most possible words which represent the drug name. The scenarios are based on the characteristics of training data, that is, drug words distribution that is usually assumed to have a smaller frequency of appearance in the dataset sentences. The drug name candidate selections are as follows. In the first case, all test dataset is taken. In the second case, 2/3 of all test dataset is selected. In the third case, x/y ($x < y$) of the test dataset (where x and y are arbitrary integer numbers) are selected after clustering the test dataset into y clusters.

In the third experiment, based on the characteristics of the resulting word vectors of the trained word embedding, we formulate a sequence data representation applied to RNN-LSTM. We used the Euclidian distance of the current input to the previous input as an additional feature besides its vector of words. In this study, the vector of words is provided by word embedding methods proposed by Mikolov et al. [13].

Our main important contributions in this study are

(1) the new data representation techniques which do not require any external knowledge nor handcrafted features,

(2) the drug extraction techniques based on the words distribution contained in the training data.

Our proposed method is evaluated on DrugBank and MedLine medical open dataset obtained from SemEval 2013 Competition task 9.1; see https://www.cs.york.ac.uk/semeval-2013/task9/, which is also used by [11, 12, 14]. The format of both medical texts is in English where some sentences contain drug name entities. In extracting drug entity names from the dataset, our data representation techniques give the best performance with F-score values 0.687 for MLP, 0.6700 for DBN, and 0.682 for SAE, whereas the third technique with LSTM gives the best F-score, that is, 0.9430. The average F-score of the third technique is 0.8645, that is, the best performance compared to the other previous methods.

By applying the data representation techniques, our proposed approach provides at least three advantages:

(1) The capability to identify multiple tokens as a single name entity

(2) The ability to deal with the absence of any external knowledge in certain languages

(3) No need to construct any additional features, such as characters type identification, orthography feature (lowercase or uppercase identification), or token position

The rest of the sections of this paper are organized as follows: Section 2 explains some previous works dealing with name entity (and drug name as well) extraction from medical text sources. The framework, approach, and methodology to overcome the challenges of drug name extraction are presented in Section 3. The section also describes dataset materials and experiment scenarios. Section 4 discusses the experiment results and its analysis while Section 5 explains the achievement, the shortcoming, and the prospects of this study. The section also describes several potential explorations for future research.

2. Related Works

The entity recognition in a biomedical text is an active research, and many methods have been proposed. For example, Pal and Gosal [9] summarize their survey on various entity recognition approaches. The approaches can be categorized into three models: dictionary based, rule-based, and learning based methods [2, 8]. A dictionary based approach uses a list of terms (term collection) to assist in predicting which targeted entity will be included in the predicted group. Although their overall precision is more accurate, their recall is poor since they anticipate less new terms. The rule-based approach defines a certain rule which describes such pattern

formation surrounding the targeted entity. This rule can be a syntactic term or lexical term. Finally, the learning approach is usually based on statistical data characteristics to build a model using machine learning techniques. The model is capable of automatic learning based on positive, neutral, and negative training data.

Drug name extraction and their classification are one of the challenges in the Semantic Evaluation Task (SemEval 2013). The best-reported performance for this challenge was 71.5% in *F*-score [15]. Until now the studies to extract drug names still continue and many approaches have been proposed. CRF-based learning is the most common method utilized in the clinical text information extraction. CRF is used by one of the best [11] participants in SemEval challenges in the clinical text (https://www.cs.york.ac.uk/semeval-2013/). As for the use of external knowledge aimed at increasing the performance, the author [11] uses ChEBI (Chemical Entities of Biological Interest), that is, a dictionary of small molecular entities. The best achieved performance is 0.57 in *F*-score (for the overall dataset).

A hybrid approach model, which combines statistical learning and dictionary based, is proposed by [16]. In their study, the author utilizes word2vec representation, CRF learning model, and DINTO, a drug ontology. With this word2vec representation, targeted drug is treated as a current token in context windows which consists of three tokens on the left and three tokens on the right. Additional features are included in the data representation such as pos tags, lemma in the windows context, and an orthography feature as uppercase, lowercase, and mixed cap. The author also used Wikipedia text as an additional resource to perform word2vec representation training. The best *F*-score value in extracting the drug name provided by the method is 0.72.

The result of CRF-based active learning, which is applied to NER BIO (Beginning, Inside, Output) annotation token for extracting name entity in the clinical text, is presented in [17]. The framework of this active learning approach is a sequential process: initial model generation, querying, training, and iteration. The CRF Algorithm BIO approach was also studied by Ben Abacha et al. [14]. The features for the CRF algorithm are formulated based on token and linguistics feature and semantic feature. The best *F*-score achieved by this proposed method is 0.72.

Korkontzelos et al. studied a combination of aggregated classifier, maximum entropy-multinomial classifier, and handcrafted feature to extract drug entity [4]. They classified drug and nondrug based on the token features formulation such as tokens windows, the current token, and 8 other handcrafted features.

Another approach for discovering valuable information from clinical text data that adopts event-location extraction model was examined by Bjorne et al. [12]. They use an SVM classifier to predict drug or nondrug entity which is applied to DrugBank dataset. The best performance achieved by their method is 0.6 in *F*-score. The drawback of their approach is that it only deals with a single token drug name.

To overcome the ambiguity problem in NER mined from a medical corpus, a segment representation method has also been proposed by Keretna et al. [8]. Their approach treats each word as belonging to three classes, that is, NE, not NE, and an ambiguous class. The ambiguity of the class member is determined by identifying whether the word appears in more than one context or not. If so, this word falls into the ambiguous class. After three class segments are found, each word is then applied to the classifier learning. Related to their approach, in our previous work, we propose pattern learning that utilizes the regular expression surrounding drug names and their compounds [18]. The performance of our method is quite good with the average *F*-score being 0.81 but has a limitation in dealing with more unstructured text data.

In summarizing the related previous works on drug name entity extraction, we noted some drawbacks which need to be addressed. In general, almost all state-of-the-art methods work based on ad hoc external knowledge which is not always available. The requirement of the handcrafted feature is another difficult constraint since not all datasets contain such feature. An additional challenge that remained unsolved by the previous works is the problem of multiple tokens representation for a single drug name. This study proposes a new data representation technique to handle those challenges.

Our proposed method is based only on the data distribution pattern and vector of words characteristics, so there is no need for external knowledge nor additional handcrafted features. To overcome the multiple tokens problem, we propose a new technique which treats a target entity as a set of tokens (a tuple) at once rather than treating the target entity as a single token surrounded by other tokens such as those used by [16] or [19]. By addressing a set of the tokens as a single sample, our proposed method can predict whether a set of tokens is a drug name or not. In our first experiment, we evaluate the first and second data representation techniques and apply MLP learning model. In our second scenario, we choose the the second technique which gave the best result with MLP and apply it to two different machine learning methods: DBN and SAE. In our third experiment, we examined the third data representation technique which utilizes the Euclidian distance between successive words in a certain sentence of medical text. The third data representation is then fed into an LSTM model. Based on the resulting *F*-score value, the second experiment gives the best performance.

3. Method and Material

3.1. Framework. In this study, using the word2vec value characteristics, we conducted three experiments based on different data representation techniques. The first and second experiment examine conventional tuple data representation, whereas the third experiment examines sequence data representation. We describe the organization of these three experiments in this section. In general, the proposed method to extract drug name entities in this study consists of two main phases. The first phase is a data representation to formulate the feature representation. In the second phase, model training, testing, and their evaluation are then conducted to evaluate the performance of the proposed method.

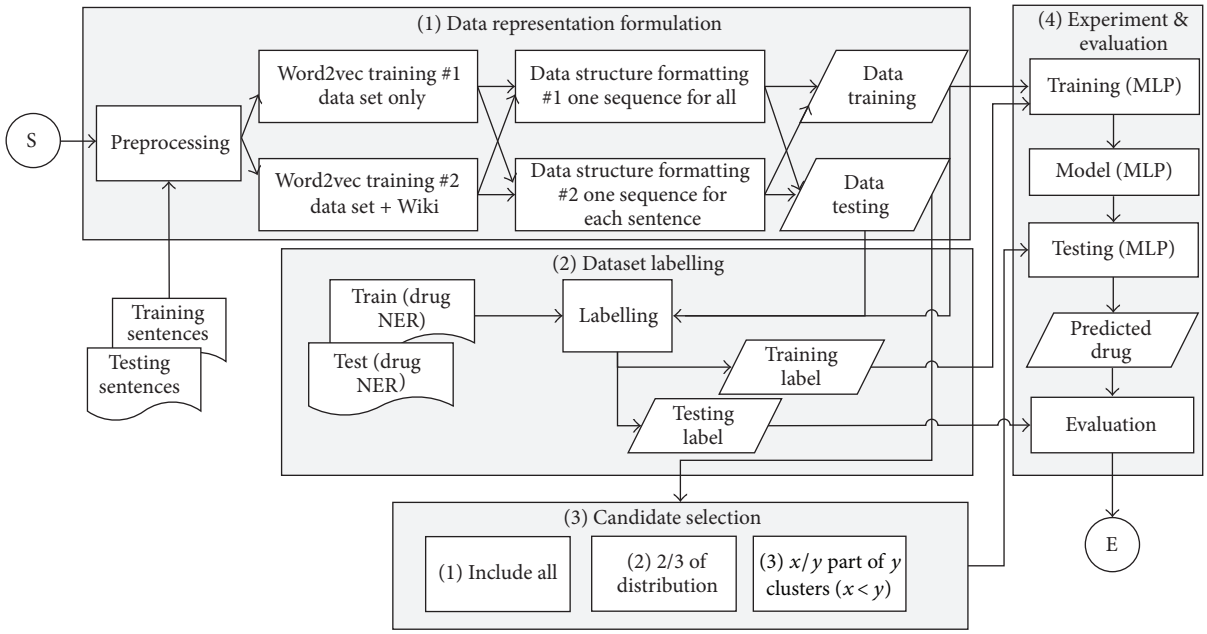


Figure 1: Proposed approach framework of the first experiment.

The proposed method of the first experiment consists of 4 steps (see Figure 1). The first step is a data representation formulation. The output of the first step is the tuples of training and testing dataset. The second step is dataset labelling which is applied to both testing and training data. The step provides the label of each tuple. The third step is the candidate selection which is performed to minimize the noises since the actual drug target quantity is far less compared to nondrug name. In the last step, we performed the experiment with MLP-NN model and its result evaluation. The detailed explanation of each step is explained in Sections 3.4, 3.6, and 3.9, whereas Sections 3.2 and 3.3 describe training data analysis as the foundation of this proposed method. As a part of the first experiment, we also evaluate the impact of the usage of the Euclidean distance average as the model's regularization. This regularization term is described in Section 3.7.1.

The framework of the second experiment which involves DBN and SAE learning model to the second data representation technique is illustrated in Figure 2. In general, the steps of the second experiment are similar to the first one, with its differences being the data representation used and the learning model involved. In the second experiment, the second technique is used only with DBN and SAE as the learning model.

The framework of the third experiment using the LSTM is illustrated in Figure 3. There are tree steps in the third experiment. The first step is sequence data representation formulation which provides both sequence training data and testing data. The second step is data labelling which generates the label of training and testing data. LSTM experiment and its result evaluation are performed in the third step. The detailed description of these three steps is presented in Sections 3.4, 3.4.3, and 3.9 as well.

3.2. Training Data Analysis. Each of the sentences in the dataset contains four data types, that is, drug, group, brand, and drug-n. If the sentence contains none of those four types, the type value is null. In the study, we extracted drug and drug-n. Overall in both DrugBank and MedLine datasets, the quantity of drug name target is far less compared to the nondrug target. Segura-Bedmar et al. [15] present the first basic statistics of the dataset. A more detailed exploration regarding token distribution in the training dataset is described in this section. The MedLine sentences training dataset contains 25.783 single tokens, which consist of 4.003 unique tokens. Those tokens distributions are not uniform but are dominated by a small part of some unique tokens. If all of the unique tokens are arranged and ranked based on the most frequent appearances in the sentences, the quartile distribution will have the following result presented in Figure 4. $Q1$ represents token numbers 1 to 1001 whose total of frequency is 20.688. $Q2$ represents token numbers 1002 to 2002 whose total of frequency is 2.849. $Q3$ represents token numbers 2003 to 3002 whose total of frequency is 1.264, and $Q4$ represents token numbers 3003 to 4003 whose total of frequency is 1.000. The figure shows that the majority of appearances are dominated by only a small amount of the total tokens.

Further analysis of the dataset tokens shows that most of the drug names of the targeted token rarely appear in the dataset. When we divide those token collections into three partitions based on their sum of frequency, as presented in Table 1, it is shown that all of the drug name entities targeted are contained in 2/3 part with less frequent appearances of each token (a unique token in the same sum of frequency). A similar pattern of training data token distribution also emerged in the DrugBank dataset as illustrated in Figure 5 and Table 2. When we look into specific token distributions,

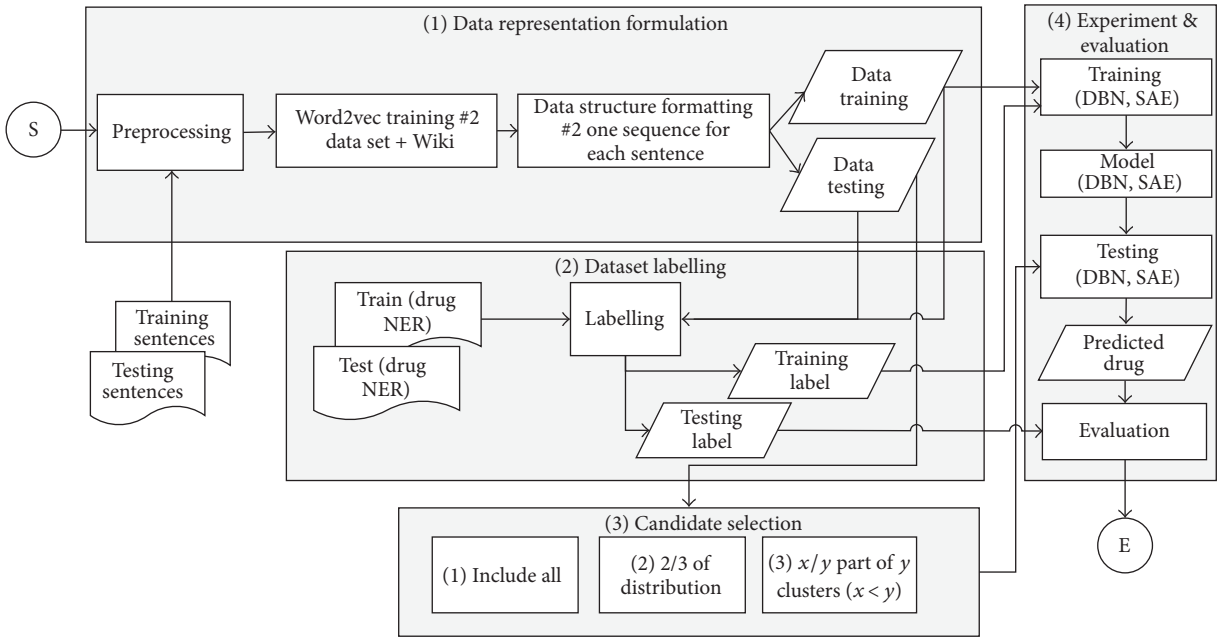


Figure 2: Proposed approach framework of the second experiment.

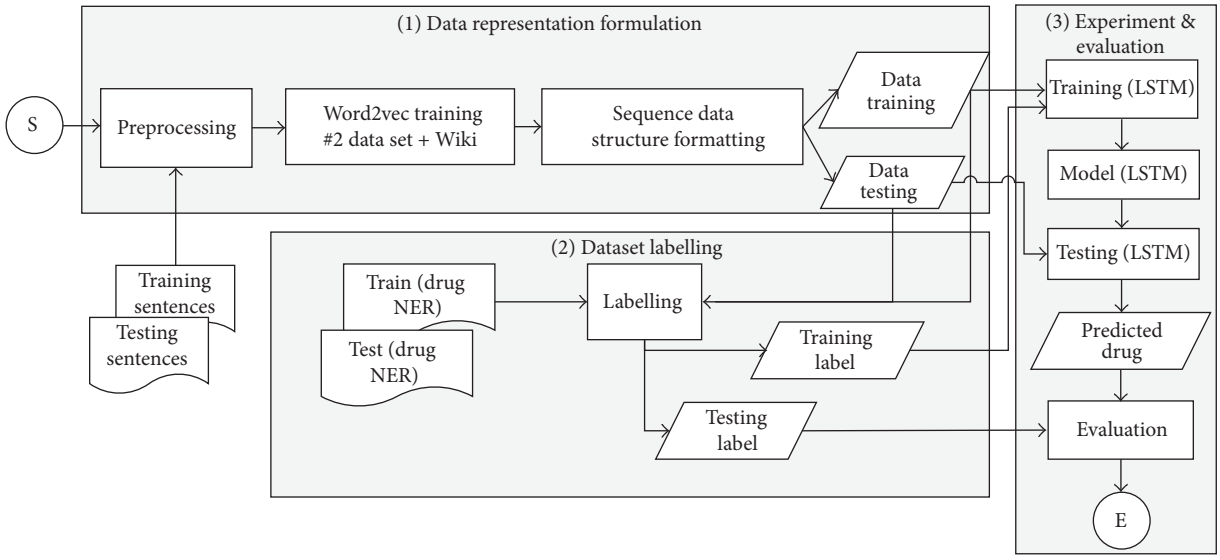


Figure 3: Proposed approach framework of the third experiment.

the position of most of the drug name targets is in the third part, since the most frequently appearing words in the first and the second parts are the most common words such as stop words ("of", "the", "a", "end", "to", "where", "as", "from", and such kind of words) and common words in medical domain such as "administrator", "patient", "effect", and "dose".

3.3. Word Embedding Analysis. To represent the dataset we utilized the word embedding model proposed by Mikolov et al. [13]. We treated all of the sentences as a corpus after the training dataset and testing dataset were combined. The used word2vec training model was the CBOW (Continuous Bag Of Words) model with context window length 5 and the vector dimension 100. The result of the word2vec training is the representation of word in 100 dimension row vectors. Based on the row vector, the similarities or dissimilarities between words can be estimated. The description below is the analysis summary of word2vec representation result which is used as a base reference for the data representation technique and the experiment scenarios. By taking some sample of drug targets and nondrug vector representation, it is shown that drug word has more similarities (cosine distance) to another drug than to nondrug and vice versa. Some of those samples are illustrated in Table 3. We also computed the Euclidean

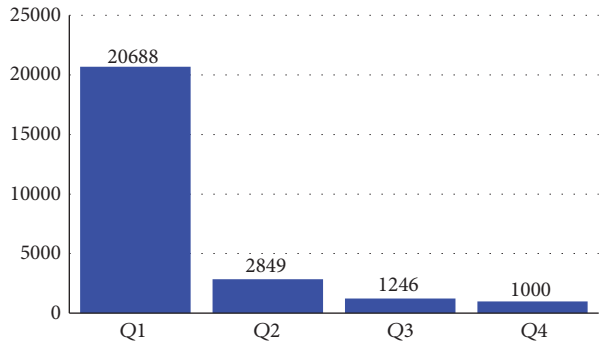


Figure 4: Distribution of MedLine train dataset token.

Table 1: The frequency distribution and drug target token position, MedLine.

1/3#	Σ sample	Σ frequency	Σ single token of drug entity
1	28	8,661	—
2	410	8,510	50
3	3,563	8,612	262

Table 2: The frequency distribution and drug target token position, DrugBank.

1/3#	Σ sample	Σ frequency	Σ single token of drug entity
1	27	33,538	—
2	332	33,463	33
3	5,501	33,351	920

distance between all of the words. Table 4 shows the average of Euclidean distance and cosine distance between drug-drug, drug-nondrug, and nondrug-nondrug. These values of the average distance show us that, intuitively, it is feasible to group the collection of the words into drug group and nondrug group based on their vector representations value.

3.4. Feature Representation, Data Formatting, and Data Labelling. Based on the training data and word embedding analysis, we formulate the feature representation and its data formatting. In the first and second techniques, we try to overcome the multiple tokens drawback left unsolved in [12] by formatting single input data as an N-gram model with $N = 5$ (one tuple piece of data consists of 5 tokens) to accommodate the maximum token which acts as a single drug entity target name. The tuples were provided from the sentences of both training and testing data. Thus, we have a set of tuples of training data and a set of tuples of testing data. Each tuple was treated as a single input.

To identify a single input, whether it is a nondrug or drug target, we use a multiclassification approach which classifies the single input into one of six classes. Class 1 represents nondrug whereas the other classes represent drug target which also identified how many tokens (words) perform the drug target. To identify which class a certain tuple belongs to the following is determined: The drug tuple is the tuple whose first token (token-1) is the drug type. If token-1 is not a drug,

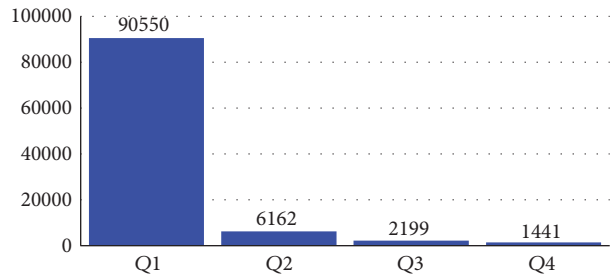


Figure 5: Distribution of DrugBank train dataset token.

regardless of whatever the rest of the 4 tokens are, then the tuple is classified as no drug. This kind of tuple is identified as class 1. If token-1 is a drug and token-2 is not a drug, regardless of the last 3 tokens, the tuple will be identified as class 2 and so on.

Since we only extracted the drug entity, we ignored the other token types, whether it is a group, brand, or another common token. To provide the label of each tuple, we only use the drug and drug-n types as the tuple reference list. In general, if the sequence of token in each tuple in dataset contains the sequence which is exactly the same with one of tuple reference list members, then the tuple in dataset is identified as drug entity. The detail of the algorithm used to provide the label of each tuple in both training data and testing data is described in Algorithm 1.

We proposed two techniques in constructing the tuple set of the sentences. The first technique treats all sentences as one sequence, whereas in the second technique, each sentence is processed as one sequence. The first and the second techniques are evaluated with MLP, DBN, and SAE model. The third technique treats the sentences of dataset as a sequence where the occurrence of the current token is influenced by the previous one. By treating the sentence as a sequence not only in the data representation but also in the classification and recognition process, the most suitable model to be used is RNN. We applied RNN-LSTM to the third technique.

3.4.1. First Technique. The first dataset formatting (one sequence for all sentences) is performed as follows. In the first step, all sentences in the dataset are formatted as a token sequence. Let the token sequence be

$$t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 \cdots t_n \quad (1)$$

with n being number of tokens in the sequences; then the dataset format will be

$$t_1 t_2 t_3 t_4 t_5; t_2 t_3 t_4 t_5 t_6; \cdots t_{n-4} t_{n-3} t_{n-2} t_{n-1} t_n. \quad (2)$$

A sample of sentences and their drug names are presented in Table 5. Taken from DrugBank training data Table 5 is the raw data of 3 samples with three relevant fields, that is, sentences, character drug position, and the drug name. Table 6 illustrates a portion of the dataset and its label as the result of the raw data in Table 5. Referring to the drug-n name field in the dataset, dataset number 6 is identified as a drug,

Table 3: Some of the cosine distance similarities between two kinds of words.

Word 1	Word 2	Similarities (cosine dist)	Remark
dilantin	tegretol	0.75135758	drug-drug
phenytoin	dilantin	0.62360351	drug-drug
phenytoin	tegretol	0.51322415	drug-drug
cholestyramine	dilantin	0.24557819	drug-drug
cholestyramine	phenytoin	0.23701277	drug-drug
administration	patients	0.20459694	non-drug - non-drug
tegretol	may	0.11605539	drug - non-drug
cholestyramine	patients	0.08827197	drug - non-drug
evaluated	end	0.07379115	non-drug - non-drug
within	controlled	0.06111103	non-drug - non-drug
cholestyramine	evaluated	0.04024139	drug - non-drug
dilantin	end	0.02234770	drug - non-drug

Table 4: The average of Euclidean distance and cosine similarities between groups of words.

Word group	Euclidean dist. avg	Cosine dist. avg
drug - non-drug	0.096113798	0.194855980
non-drug - non-drug	0.094824332	0.604091044
drug-drug	0.093840800	0.617929002

Table 5: Sample of DrugBank sentences and their drug name target.

Sentence	Drug position	Drug name
modification of surface histidine residues abolishes the cytotoxic activity of clostridium difficile toxin a	79–107	clostridium difficile toxin a
antimicrobial activity of ganoderma lucidum extract alone and in combination with some antibiotics.	26–50	ganoderma lucidum extract
on the other hand, surprisingly, green tea gallocatechins, (−)-epigallocatechin-3-o-gallate and theasinensin a, potently enhanced the promoter activity (182 and 247% activity at 1 microm, resp.).	33–56	green tea gallocatechins

whereas the others are classified as a nondrug entity. The complete label illustration of the dataset provided by the first technique is presented in Table 7. As described in Section 3.4, the value of vector dimension for each token is 100. Therefore, for single data, it is represented as $100 * 5 = 500$ lengths of a one-dimensional vector.

3.4.2. Second Technique. The second technique is used for treating one sequence that comes from each sentence of the dataset. With this treatment, we added special characters $*$, as padding, to the last part of the token when its dataset length is less than 5. By applying the second technique the first sentence of the sample provided a dataset as illustrated in Table 8.

3.4.3. Third Technique. Naturally, the NLP sentence is a sequence in which the occurrence of the current word is conditioned by the previous one. Based on the word2vec value analysis, it is shown that intuitively we can separate the drug word and nondrug word by their Euclidean distance. Therefore, we used the Euclidean distance between the current words with the previous one to represent the influence. Thus, each current input x_i is represented by $[xv_i xd_i]$ which is the concatenation of word2vec value xv_i and its Euclidian distance to the previous one, xd_i. Each x is the row vector with the dimension length being 200, the first 100 values are its word2vector, and the rest of all 100 values are the Euclidian distance to the previous. For the first word all values of xd_i are 0. With the LSTM model, the task to extract the drug name from the medical data text is the binary classification applied to each word of the sentence. We formulate the word sequence and its class as described in Table 9. In this experiment, each word that represents the drug name is identified as class 1, such as "plenaxis", "cytochrome", and "p-450", whereas the other words are identified by class 0.

3.5. Wiki Sources. In this study we also utilize Wikipedia as the additional text sources in word2vec training as used by [16]. The Wiki text addition is used to evaluate the impact of the training data volume in improving the quality of word's vector.

3.6. Candidates Selection. The tokens as the drug entities target are only a tiny part of the total tokens. In MedLine dataset, 171 of 2.000 tokens (less than 10%) are drugs, whereas in DrugBank, the number of drug tokens is 180 of 5.252 [15]. So the major part of these tokens is nondrug and other noises such as a stop word and special or numerical characters. Based on this fact, we propose a candidate selection step to eliminate those noises. We examine two mechanisms in the candidate selection. The first is based on token distribution. The second is formed by selecting x/y part of the clustering result of data test. In the first scenario, we only used 2/3 of the token, which appears in the lower 2/3 part of the total token. This is presented in Tables 1 and 2. However, in the second

```
Result: Labelled dataset
Input: array of tuple, array of drug;
output: array of label {Array of drug contains list of drug and drug-n only};
label[] <= 1 Initialization;
for each t in tuple do
    for each d in drug do
        if length (d) = 1 then
            if t[1] = d[1] then
                //match 1 token drug;
                label <= 2, break, exit from for each d in drug;
            else
            end
        else
            if length (d) = 2 then
                if t[1] = d[1] and t[2] = d[2] then
                    //match 2 tokens drug;
                    label <= 3, break, exit from for each d in drug;
                else
                end
            else
                if length (d) = 3 then
                    if t[1] = d[1] and t[2] = d[2] and t[3] = d[3] then
                        label <= 4, break, exit from for each d in drug;
                    else
                    end
                else
                    if length (d) = 4 then
                        if t[1] = d[1] and t[2] = d[2] and t[3] = d[3] and t[4] = d[4] then
                            label <= 5, break, exit from for each d in drug;
                        else
                        end
                    else
                        if length (d) = 5 then
                            if t[1] = d[1] and t[2] = d[2] and t[3] = d[3] and t[4] = d[4] and t[5] = d[5] then
                                label <= 6, break, exit from for each d in drug;
                            else
                            end
                        else
                        end
                    end
                end
            end
        end
    end
end
```

Algorithm 1: Dataset labelling.

Table 6: A portion of the dataset formulation as the results of DrugBank sample with first technique.

Dataset number	Token-1	Token-2	Token-3	Token-4	Token-5	Label
1	modification	of	surface	histidine	residues	1
2	of	surface	histidine	residues	abolishes	1
3	surface	histidine	residues	abolishes	the	1
4	histidine	residues	abolishes	the	cytotoxic	1
5	the	cytotoxic	activity	of	clostridium	1
6	*clostridium*	*difficile*	*toxin*	*a*	antimicrobial	5
7	difficile	toxin	a	antimicrobial	activity	1

Table 7: First technique of data representation and its label.

Token-1	Token-2	Token-3	Token-4	Token-5	Label
"plenaxis"	"were"	"performed"	"cytochrome"	"p-450"	2
"testosterone"	"concentrations"	"just"	"prior"	"to"	2
"beta-adrenergic"	*"antagonists"*	"and"	"alpha-adrenergic"	"stimulants,"	3
"carbonic"	*"anhydrase"*	"inhibitors,"	"concomitant"	"use"	3
"sodium"	*"polystyrene"*	*"sulfonate"*	"should"	"be"	4
"sodium"	*"acid"*	*"phosphate"*	"such"	"as"	4
"clostridium"	*"difficile"*	*"toxin"*	*"a"*	"—"	5
"nonsteroidal"	*"anti"*	*"inflammatory"*	*"drugs"*	"and"	5
"casein"	*"phosphopeptide-amorphous"*	*"calcium"*	*"phosphate"*	*"complex"*	6
"studies"	"with"	"plenaxis"	"were"	"performed."	1
"were"	"performed."	"cytochrome"	"p-450"	"is"	1

Table 8: Second technique of data representation and its label.

Token-1	Token-2	Token-3	Token-4	Token-5	Label
"modification"	"of"	"surface"	"histidine"	"residues"	1
"of"	"surface"	"histidine"	"residues"	"abolishes"	1
surface	histidine	residues	abolishes	the	1
"histidine"	"residues"	"abolishes"	"the"	"cytotoxic"	1
"the"	"cytotoxic"	"activity"	"of"	"clostridium"	1
"clostridium"	*"difficile"*	*"toxin"*	"a"	"∗"	5
"difficile"	"toxin"	"a"	"∗"	"∗"	1
"a"	"atoxin"	"∗"	"∗"	"∗"	1
"toxic"	"∗"	"∗"	"∗"	"∗"	1

mechanism we selected x/y ($x < y$) which is a part of total token after the tokens are clustered into y clusters.

3.7. Overview of NN Model

3.7.1. MLP. In the first experiment, we used multilayer perceptron NN to train the model and evaluate the performance [20]. Given a training set of m examples, then the overall cost function can be defined as

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J\left(W,b;x^i,y^i\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{sl}\sum_{j=1}^{sl-1}\left(W_{j_i}^{(l)}\right)^2,$$
$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{wb}\left(x^{(i)}\right)-y^i\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{sl}\sum_{j=1}^{sl-1}\left(W_{j_i}^{(l)}\right)^2. \quad (3)$$

In the definition of $J(W,b)$, the first term is an average sum-of-squares error term, whereas the second term is a regularization term which is also called a weight decay term. In this experiment we use three kinds of regularization: #0, L0 with $\lambda = 0$, #1, L1 with $\lambda = 1$, and #2 with $\lambda =$ the average of Euclidean distance. We computed the L2's λ based on the word embedding vector analysis that drug target and nondrug can be distinguished by looking at their Euclidean distance. Thus, for L2 regularization, the parameter is calculated as

$$\lambda = \frac{1}{n*(n-1)}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\operatorname{dist}\left(x^i,x^j\right), \quad (4)$$

where $\operatorname{dist}(x^i,x^j)$ is the Euclidean distance of x^i and x^j.

The model training and testing are implemented by modifying the code from [21] which can be downloaded at https://github.com/rasmusbergpalm/DeepLearnToolbox.

3.7.2. DBN. DBN is a learning model composed of two or more stacked RBMs [22, 23]. An RBM is an undirected graph learning model which associates with Markov Random Fields (MRF). In the DBN, the RBM acts as feature extractor where the pretraining process provides initial weights values to be fine-tuned in the discriminative process in the last layer. The last layer may be formed by logistic regression or any standard discriminative classifiers [23]. RBM was originally developed for binary data observation [24, 25]. It is a popular type of unsupervised model for binary data [26, 27]. Some derivatives of RBM models are also proposed to tackle continuous/real values suggested in [28, 29].

Table 9: Third technique of data representation and its label.

Sent.#1	Class	0	0	0	0	1	0	0
	Word	"drug"	"interaction"	"studies"	"with"	"plenaxis"	"were"	"performed"
Sent.#2	Class	1	1	0	0	0	0	0
	Word	"cytochrome"	"p-450"	"is"	"not"	"known"	"in"	"the"

3.7.3. SAE. An autoencoder (AE) neural network is one of the unsupervised learning algorithms. The NN tries to learn a function $h(w, x) \approx x$. The autoencoder NN architecture also consists of input, hidden, and output layers. The particular characteristic of the autoencoder is that the target output is similar to the input. The interesting structure of the data is estimated by applying a certain constraint to the network, which limits the number of hidden units. However, when the number of hidden units has to be larger, it can be imposed with sparsity constraints on the hidden units [30]. The sparsity constraint is used to enforce the average value of hidden unit activation constrained to a certain value. As used in the DBN model, after we trained the SAE, the trained weight was used to initialize the weight of NN for the classification.

3.7.4. RNN-LSTM. RNN (Recurrent Neural Network) is an NN, which considers the previous input in determining the output of the current input. RNN is powerful when it is applied to the dataset with a sequential pattern or when the current state input depends on the previous one, such as the time series data, sentences of NLP. An LSTM network is a special kind of RNN which also consists of 3 layers, that is, an input layer, a single recurrent hidden layer, and an output layer [31]. The main innovation of LSTM is that its hidden layer consists of one or more memory blocks. Each block includes one or more memory cells. In the standard form, the inputs are connected to all of the cells and gates, whereas the cells are connected to the outputs. The gates are connected to other gates and cells in the hidden layer. The single standard LSTM is a hidden layer with input, memory cell, and output gates [32, 33].

3.8. Dataset. To validate the proposed approach, we utilized DrugBank and MedLine open dataset, which have also been used by previous researchers. Additionally, we used drug label documents from various drug producers and regulator Internet sites located in Indonesia:

(1) http://www.kalbemed.com/

(2) http://www.dechacare.com/

(3) http://infoobatindonesia.com/obat/, and

(4) http://www.pom.go.id/webreg/index.php/home/produk/01.

The drug labels are written in Bahasa Indonesia, and their common contents are drug name, drug components, indication, contraindication, dosage, and warning.

3.9. Evaluation. To evaluate the performance of the proposed method, we use common measured parameters in data mining, that is, precision, recall, and F-score. The computation formula of these parameters is as follows. Let $C = \{C_1, C_2, C_3, \ldots, C_n\}$ be a set of the extracted drug name of this method, and $K = \{K_1, K_2, K_3, \ldots, K_l\}$ is set of actual drug names in the document set D. Adopted from [18], the parameter computations formula is

$$\begin{aligned} \text{Precision}\left(K_i, C_j\right) &= \frac{(\text{TruePositive})}{(\text{TruePositive} + \text{FalsePositive})} \\ &= \frac{\left(\left\|K_i \cap C_j\right\|\right)}{\left(\left\|C_j\right\|\right)}, \\ \text{Recall}\left(K_i, C_j\right) &= \frac{(\text{TruePositive})}{(\text{TruePositive} + \text{FalseNegative})} \\ &= \frac{\left(\left\|K_i \cap C_j\right\|\right)}{\left(\left\|K_i\right\|\right)}, \end{aligned} \tag{5}$$

where $\|K_i\|$, $\|C_j\|$, and $\|K_i \cap C_j\|$ denote the number of drug names in K, in C, and in both K and C, respectively. The F-score value is computed by the following formula:

$$\begin{aligned} &F\text{-score}\left(K_i, C_j\right) \\ &\quad = \frac{\left(2 * \text{Precision}\left(K_i, C_j\right) * \text{Recall}\left(K_i, C_j\right)\right)}{(\text{TruePositive} + \text{FalsePositive})}. \end{aligned} \tag{6}$$

4. Results and Discussion

4.1. MLP Learning Performance. The following experiments are the part of the first experiment. These experiments are performed to evaluate the contribution of the three regularization settings as described in Section 3.7.1. By arranging the sentence in training dataset as 5-gram of words, the quantity of generated sample is presented in Table 10. We do training and testing of the MLP-NN learning model for all those test data compositions. The result of model performances on both datasets, that is, MedLine and DrugBank, in learning phase is shown in Figures 6 and 7. The NN learning parameters that are used for all experiments are 500 input nodes, two hidden layers where each layer has 100 nodes with sigmoid activation, and 6 output nodes with softmax function; the learning rate = 1, momentum = 0.5, and epochs = 100. We used minibatch scenario in the training with the batch size being 100. The presented errors in Figures 6 and 7 are the errors for full batch, that is, the mean errors of all minibatches.

Table 10: Dataset composition.

Dataset	Train	Test		
		All	2/3 part	Cluster
MedLine	26,500	10,360	6,673	5,783
DrugBank	100,100	2,000	1,326	1,933

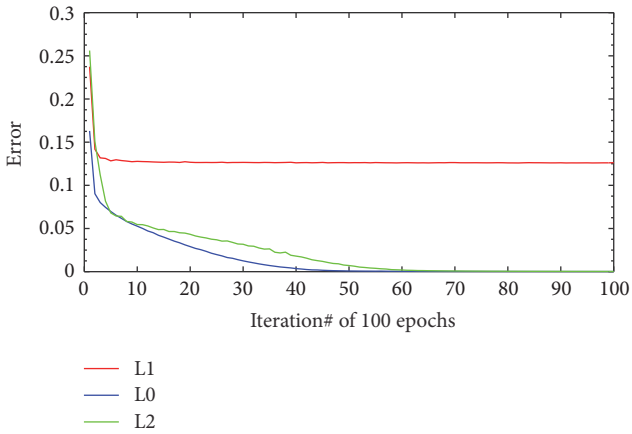


Figure 6: Full batch training error of MedLine dataset.

The learning model performance shows different patterns between MedLine and DrugBank datasets. For both datasets, L1 regularization tends to stabilize in the lower iteration and its training error performance is always less than L0 or L2. The L0 and L2 training error performance pattern, however, shows a slight different behavior between MedLine and DrugBank. For the MedLine dataset, L0 and L2 produce different results for some of the iterations. Nevertheless, the training error performance of L0 and L2 for DrugBank is almost the same in every iteration. Different pattern results are probably due to the variation in the quantity of training data. As illustrated in Table 10, the volume of DrugBank training data is almost four times the volume of the MedLine dataset. It can be concluded that, for larger dataset, the contribution of L2 regularization setting is not too significant in achieving better performance. For smaller dataset (MedLine), however, the performance is better even after only few iterations.

4.2. Open Dataset Performance. In Tables 11, 12, 13, and 14, numbering (1), numbering (2), and numbering (3) in the most left column indicate the candidate selection technique with

(i) (1): all data tests being selected;

(ii) (2): 2/3 part of data test being selected;

(iii) (3): 2/3 part of 3 clusters for MedLine or 3/4 part of 4 clusters for DrugBank.

4.2.1. MLP-NN Performance. In this first experiment, for two data representation techniques and three candidate selection scenarios, we have six experiment scenarios. The result

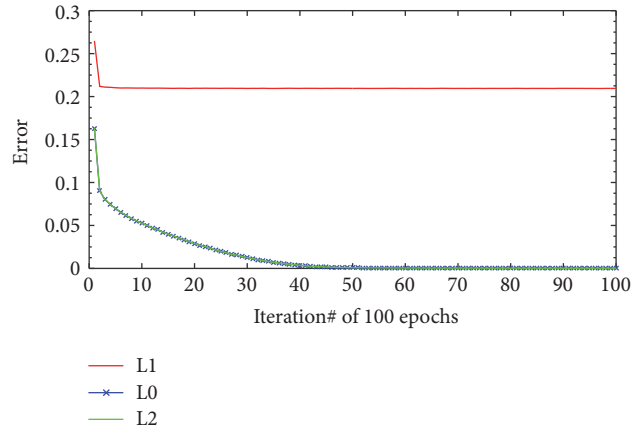


Figure 7: Full batch training error of DrugBank dataset.

of the experiment which applies the first data representation technique and three candidate selection scenarios is presented in Table 11. In computing the F-score, we only select the predicted target which is provided by the lowest error (the minimum one). For MedLine dataset, the best performance is shown by L2 regularization setting where the error is 0.041818, in third candidate selection scenario with F-score 0.439516, whereas the DrugBank is achieved together by L0 and L1 regularization setting, with an error test of 0.0802; in second candidate selection scenario, the F-score was 0.641745. Overall, it can be concluded that DrugBank experiments give the best F-score performance. The candidate selection scenarios also contributed to improving the performance, as we found that, for both of MedLine and DrugBank, the best achievement is provided by the second and third scenarios, respectively.

The next experimental scenario in the first experiment is performed to evaluate the impact of the data representation technique and the addition of Wiki source in word2vec training. The results are presented in Tables 12 and 13. According to the obtained results presented in Table 11, the L0 regularization gives the best F-score. Hence, accordingly we only used the L0 regularization for the next experimental scenario. Table 12 presents the impact of the data representation technique. Looking at the F-score, the second technique gives better results for both datasets, that is, the MedLine and DrugBank.

Table 13 shows the result of adding the Wiki source into word2vec training in providing the vector of word representation. These results confirm that the addition of training data will improve the performance. It might be due to the fact that most of the targeted tokens such as drug name are uncommon words, whereas the words that are used in Wiki's sentence are commonly used words. Hence, the addition of commonly used words will make the difference between drug token and the nondrug token (the commonly used token) become greater. For the MLP-NN experimental results, the 4th scenario, that is, the second data representation with 2/3 partition data selection in DrugBank dataset, provides the best performance with 0.684646757 in F-score.

Table 11: The F-score performances of three of scenarios experiments.

MedLine	Prec	Rec	F-score	Lx	Error test
(1)	0.3564	0.5450	0.4310	L0	0.0305
(2)	0.3806	0.5023	0.4331	L1, L2	0.0432
(3)	0.3773	0.5266	0.4395	L2	0.0418
DrugBank	Prec	Rec	F-score	Lx	Error test
(1)	0.6312	0.5372	0.5805	L0	0.07900
(2)	0.6438	0.6398	0.6417	L0, L2	0.0802
(3)	0.6305	0.5380	0.5806	L0	0.0776

Table 12: The F-score performance as an impact of data representation technique.

Dataset	(1) One seq. of all sentences			(2) One seq. of each sentence		
MedLine	Prec	Rec	F-score	Prec	Rec	F-score
(1)	0.3564	0.5450	0.4310	0.6515	0.6220	0.6364
(2)	0.3806	0.5023	0.4331	0.6119	0.7377	**0.6689**
(3)	0.3772	0.5266	**0.4395**	0.6143	0.656873	0.6348
DrugBank	Prec	Rec	F-score	Prec	Rec	F-score
(1)	0.6438	0.5337	0.5836	0.7143	0.4962	0.5856
(2)	0.6438	0.6398	**0.6418**	0.7182	0.5804	**0.6420**
(3)	0.6306	0.5380	0.5807	0.5974	0.5476	0.5714

4.2.2. DBN and SAE Performance. In the second experiment, which involves DBN and SAE learning model, we only use the experiment scenario that gives the best results in the first experiment. The best experiment scenario uses the second data representation technique with Wiki text as an additional source in the word2vec training step.

In the DBN experiment, we use two stacked RBMs with 500 nodes of visible unit and 100 nodes of the hidden layer for the first and also the second RBMs. The used learning parameters are as follows: momentum = 0 and alpha = 1. We used minibatch scenario in the training, with the batch size of 100. As for RBM constraints, the range of input data value is restricted to $[0 \cdots 1]$ as the original RBM, which is developed for binary data type, whereas the range of vector of word value is $[-1 \cdots 1]$. So we normalize the data value into $[0 \cdots 1]$ range before performing the RBM training. In the last layer of DBN, we use one layer of MLP with 100 hidden nodes and 6 output nodes with softmax output function as classifier.

The used SAE architecture is two stacked AEs with the following nodes configuration. The first AE has 500 units of visible unit, 100 hidden layers, and 500 output layers. The second AE has 100 nodes of visible unit, 100 nodes of hidden unit, and 100 nodes of output unit. The used learning parameters for first SAE and the second SAE, respectively, are as follows: activation function = sigmoid and tanh; learning rate = 1 and 2; momentum = 0.5 and 0.5; sparsity target = 0.05 and 0.05. The batch size of 100 is set for both of AEs. In the SAE experiment, we use the same discriminative layer as DBN, that is, one layer MLP with 100 hidden nodes and 6 output nodes with softmax activation function.

The experiments results are presented in Table 14. There is a difference in performances when using the MedLine and the DrugBank datasets when feeding them into MLP, DBN, and SAE models. The best results for the MedLine dataset are obtained when using the SAE. For the DrugBank, the MLP gives the best results. The DBN gives lower average performance for both datasets. The lower performance is probably due to the normalization on the word vector value to $[0 \cdots 1]$, whereas their original value range is in fact between $[-1 \cdots 1]$. The best performance for all experiments 1 and 2 is given by SAE, with the second scenario of candidate selection as described in Section 3.6. Its F-score is 0.686192469.

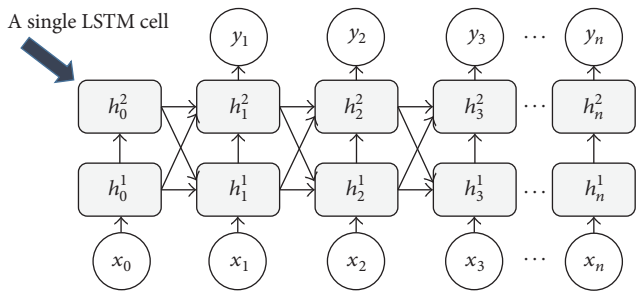


Figure 8: The global LSTM network.

4.2.3. LSTM Performance. The global LSTM network used is presented in Figure 8. Each single LSTM block consists of two stacked hidden layers and one input node with each input dimension being 200 as described in Section 3.4.3. All hidden layers are fully connected. We used sigmoid as an output activation function, which is the most suitable for binary classification. We implemented a peepholes connection LSTM variant where its gate layers look at the cell state [34]. In addition to implementing the peepholes connection,

Table 13: The *F*-score performances as an impact of the Wiki addition of word2vec training data.

Dataset	(1) One seq. of all sentences			(2) One seq. of each sentence		
MedLine	Prec	Rec	*F*-score	Prec	Rec	*F*-score
(1)	0.5661	0.4582	0.5065	0.614	0.6495	0.6336
(2)	0.5661	0.4946	**0.5279**	0.5972	0.7454	0.6631
(3)	0.5714	0.4462	0.5011	0.6193	0.6927	**0.6540**
DrugBank	Prec	Rec	*F*-score	Prec	Rec	*F*-score
(1)	0.6778	0.5460	0.6047	0.6973	0.6107	0.6511
(2)	0.6776	0.6124	**0.6433**	0.6961	0.6736	**0.6846**
(3)	0.7173	0.5574	0.6273	0.6976	0.6193	0.6561

Table 14: Experimental results of three NN models.

Dataset	MLP			DBN			SAE		
MedLine	Prec	Rec	*F*-score	Prec	Rec	*F*-score	Prec	Rec	*F*-score
(1)	0.6515	0.6220	0.6364	0.5464	0.6866	0.6085	0.6728	0.6214	0.6461
(2)	0.5972	0.7454	0.6631	0.6119	0.7377	0.6689	0.6504	0.7261	**0.6862**
(3)	0.6193	0.6927	0.6540	0.6139	0.6575	0.6350	0.6738	0.6518	0.6626
Average	0.6227	0.6867	0.6512	0.5907	0.6939	0.6375	0.6657	0.6665	0.6650
DrugBank	Prec	Rec	*F*-score	Prec	Rec	*F*-score	Prec	Rec	*F*-score
(1)	0.6973	0.6107	0.6512	0.6952	0.5847	0.6352	0.6081	0.6036	0.6059
(2)	0.6961	0.6736	**0.6847**	0.6937	0.6479	0.6700	0.6836	0.6768	0.6802
(3)	0.6976	0.6193	0.6561	0.6968	0.5929	0.6406	0.6033	0.6050	0.6042
Average	0.6970	0.6345	0.664	0.6952	0.6085	0.6486	0.6317	0.6285	0.6301

we also use a couple of forget and input gates. The detailed single LSTM architecture and each gate formula computation can be referred to in [33].

The LSTM experiments were implemented with several different parameter settings. Their results presented in this section are the best among all our experiments. Each piece of input data consists of two components, its word vector value and its Euclidian distance to the previous input data. In treating both input data components, we adapt the Adding Problem Experiment as presented in [35]. We use the Jannlab tools [36] with some modifications in the part of entry to conform with our data settings.

The best achieved performance is obtained with LSTM block architecture of one node input layer, two nodes' hidden layer, and one node output layer. The used parameters are learning rate = 0.001, momentum = 0.9, epoch = 30, and input dimension = 200, with the time sequence frame set to 2. The complete treatment of drug sentence as a sequence both in representation and in recognition, to extract the drug name entities, is the best technique, as shown by *F*-score performance in Table 15.

As described in previous work section, there are many approaches related to drug extraction that have been proposed. Most of them utilize certain external knowledge to achieve the extraction objective. Table 16 summarizes their *F*-score performance. Among the state-of-the-art techniques, our third data representation technique applied to the LSTM model is outperforming. Also, our proposed method does not require any external knowledge.

Table 15: The *F*-score performance of third data representation technique with RNN-LSTM.

	Prec	Rec	*F*-score
MedLine	1	0.6474	0.7859
DrugBank	1	0.8921	0.9430
Average			0.8645

4.3. Drug Label Dataset Performance. As additional experiment, we also use Indonesian language drug label corpus to evaluate the method's performance. Regarding the Indonesian drug label, we could not find any certain external knowledge that can be used to assist the extraction of the drug name contained in the drug label. In the presence of this hindrance, we found our proposed method is more suitable than any other previous approaches. As the drug label texts are collected from various sites of drug distributors, producers, and government regulators, they do does not clearly contain training data and testing data as in DrugBanks or MedLine datasets. The other characteristics of these texts are the more structured sentences contained in the data. Although the texts are coming from various sources, all of them are similar kind of document (the drug label that might be generated by machine). After the data cleaning step (HTML tag removal, etc.), we annotated the dataset manually. The total quantity of dataset after performing the data representation step, as described in Section 3.4, is

Table 16: The *F*-score performance compared to the state of the art.

Approach	*F*-score	Remark
The Best of SemEval 2013 [15]	0.7150	—
[11]	0.5700	With external knowledge, ChEBI
[16] + Wiki	0.7200	With external knowledge, DINTO
[14]	0.7200	Additional feature, BIO
[12]	0.6000	Single token only
MLP-SentenceSequence + Wiki (average)/Ours	0.6580	Without external knowledge
DBN-SentenceSequence + Wiki (average)/Ours	0.6430	Without external knowledge
SAE-SentenceSequence + Wiki (average)/Ours	0.6480	Without external knowledge
LSTM-AllSentenceSequence + Wiki + EuclidianDistance (average)/Ours	0.8645	Without external knowledge

Table 17: The best performance of 10 executions on drug label corpus.

Iteration	Prec	Recall	*F*-score
1	0.9170	0.9667	0.9412
2	0.8849	0.9157	0.9000
3	0.9134	0.9619	0.9370
4	0.9298	0.9500	0.9398
5	0.9640	0.9570	0.9605
6	0.8857	0.9514	0.9178
7	0.9489	0.9689	0.9588
8	0.9622	0.9654	0.9638
9	0.9507	0.9601	0.9554
10	0.9516	0.9625	0.9570
Average	0.93081	0.9560	0.9431
Min	0.8849	0.9157	0.9000
Max	0.9640	0.9689	0.9638

1.046.200. In this experiment, we perform 10 times cross-validation scenario by randomly selecting 80% data for the training data and 20% data for testing.

The experimental result for drug label dataset shows that all of the candidate selection scenarios provide excellent *F*-score (above 0.9). The excellent *F*-score performance is probably due to the more structured sentences in those texts. The best results of those ten experiments are presented in Table 17.

4.4. Choosing the Best Scenario. In the first and second experiments, we studied various experiment scenarios, which involve three investigated parameters: additional Wiki source, data representation techniques, and drug target candidate selection. In general, the Wiki addition contributes to improving the *F*-score performance. The additional source in word2vec training enhances the quality of the resulting word2vec. Through the addition of common words, from Wiki, the difference between the common words and the uncommon words, that is, drug name, becomes greater (better distinguishing power).

One problem in mining drug name entity from medical text is the imbalanced quantity between drug token and other tokens [15]. Also, the targeted drug entities are only a small part of the total tokens. Thus, majority of tokens are noise. In dealing with this problem, the second and third candidate selection scenarios show their contribution to reduce the quantity of noise. Since the possibility of extracting the noises is reduced then the recall value and *F*-score value increase as well, as shown in the first and second experiments results.

The third experiment which uses LSTM model does not apply the candidate selection scenario because the input dataset is treated as sentence sequence. So the input dataset can not be randomly divided (selected) as the tuple treatment in the first and second experiments.

5. Conclusion and Future Works

This study proposes a new approach in the data representation and classification to extract drug name entities contained in the sentences of medical text documents. The suggested approach solves the problem of multiple tokens for a single entity that remained unsolved in previous studies. This study also introduces some techniques to tackle the absence of specific external knowledge. Naturally, the words contained in the sentence follow a certain sequence pattern; that is, the current word is conditioned by other previous words. Based on the sequence notion, the treatment of medical text sentences which apply the sequence NN model gives better results. In this study, we presented three data representation techniques. The first and second techniques treat the sentence as a nonsequence pattern which is evaluated with the non-sequential NN classifier (MLP, DBN, and SAE), whereas the third technique treats the sentences as a sequence to provide data that is used as the input of the sequential NN classifier, that is, LSTM. The performance of the application of LSTM models for the sequence data representation, with the average *F*-score being 0.8645, rendered the best result compared to the state of the art.

Some opportunities to improve the performance of the proposed technique are still widely opened. The first step improvement can be the incorporation of additional hand-crafted features, such as the words position, the use of capital case at the beginning of the word, and the type of character, as also used in the previous studies [16, 37]. As presented in the MLP experiments for drug label document, the proposed methods achieved excellent performance when applied to the more structured text. Thus, the effort to make the sentence of the dataset, that is, DrugBank and MedLine, to be more structured can also be elaborated. Regarding the LSTM model and the sequence data representation for the sentences of medical text, our future study will tackle the multiple entity extractions such as drug group, drug brand, and drug compounds. Another task that is potential to be solved with

the LSTM model is the drug-drug interaction extraction. Our experiments also utilize the Euclidean distance measure in addition to the word2vec features. Such addition gives a good F-score performance. The significance of embedding the Euclidean distance features, however, needs to be explored further.

Competing Interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work is supported by Higher Education Science and Technology Development Grant funded by Indonesia Ministry of Research and Higher Education Contract no. 1004/UN2.R12/HKP.05.00/2016.

References

[1] M. Sadikin and I. Wasito, "Translation and classification algorithm of FDA-Drugs to DOEN2011 class therapy to estimate drug-drug interaction," in *Proceedings of the The 2nd International Conference on Information Systems for Business Competitiveness (ICISBC '13)*, pp. 1–5, Semarang, Indonesia, 2013.

[2] H. Tang and J. Ye, "A survey for information extraction method," Tech. Rep., 2007.

[3] S. Zhang and N. Elhadad, "Unsupervised biomedical named entity recognition: experiments with clinical and biological texts," *Journal of Biomedical Informatics*, vol. 46, no. 6, pp. 1088–1098, 2013.

[4] I. Korkontzelos, D. Piliouras, A. W. Dowsey, and S. Ananiadou, "Boosting drug named entity recognition using an aggregate classifier," *Artificial Intelligence in Medicine*, vol. 65, no. 2, pp. 145–153, 2015.

[5] M. Herrero-Zazo, I. Segura-Bedmar, P. Martínez, and T. Declerck, "The DDI corpus: an annotated corpus with pharmacological substances and drug-drug interactions," *Journal of Biomedical Informatics*, vol. 46, no. 5, pp. 914–920, 2013.

[6] H. Sampathkumar, X.-W. Chen, and B. Luo, "Mining adverse drug reactions from online healthcare forums using Hidden Markov Model," *BMC Medical Informatics and Decision Making*, vol. 14, article 91, 2014.

[7] I. Segura-Bedmar, P. Martínez, and M. Segura-Bedmar, "Drug name recognition and classification in biomedical texts. A case study outlining approaches underpinning automated systems," *Drug Discovery Today*, vol. 13, no. 17-18, pp. 816–823, 2008.

[8] S. Keretna, C. P. Lim, D. Creighton, and K. B. Shaban, "Enhancing medical named entity recognition with an extended segment representation technique," *Computer Methods and Programs in Bimoedicine*, vol. 119, no. 2, pp. 88–100, 2015.

[9] G. Pal and S. Gosal, "A survey of biological entity recognition approaches," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, no. 9, 2015.

[10] S. Liu, B. Tang, Q. Chen, and X. Wang, "Drug name recognition: approaches and resources," *Information*, vol. 6, no. 4, pp. 790–810, 2015.

[11] T. Grego and F. M. Couto, "LASIGE: using conditional random fields and ChEBI ontology," in *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval '13)*, vol. 2, pp. 660–666, 2013.

[12] J. Bjorne, S. Kaewphan, and T. Salakoski, "UTurku: drug named entity recognition and drug-drug interaction extraction using SVM classification and domain knowledge," in *Proceedings of the 2nd Joint Conferernce on Lexical and Computational Semantic*, vol. 2, pp. 651–659, Atlanta, Ga, USA, 2013.

[13] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient estimation of word representations in vector space," https://arxiv.org/abs/1301.3781.

[14] A. Ben Abacha, M. F. M. Chowdhury, A. Karanasiou, Y. Mrabet, A. Lavelli, and P. Zweigenbaum, "Text mining for pharmacovigilance: using machine learning for drug name recognition and drug-drug interaction extraction and classification," *Journal of Biomedical Informatics*, vol. 58, pp. 122–132, 2015.

[15] I. Segura-Bedmar, P. Martinez, and M. Herrero-Zazo, "Semeval-2013 task 9: extraction of drug-drug interactions from biomedical texts (ddiextraction 2013)," in *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval '13)*, vol. 2, pp. 341–350, Association for Computational Linguistics, Atlanta, Georgia, USA, 2013.

[16] I. Segura-bedmar and P. Mart, "Exploring word embedding for drug name recognition," in *Proceedings of the The 6th International Workshop on Health Text Mining and Information Analysis*, pp. 64–72, Lisbon, Portugal, September 2015.

[17] Y. Chen, T. A. Lasko, Q. Mei, J. C. Denny, and H. Xu, "A study of active learning methods for named entity recognition in clinical text," *Journal of Biomedical Informatics*, vol. 58, pp. 11–18, 2015.

[18] M. Sadikin and I. Wasito, "Toward object interaction mining by starting with object extraction based on pattern learning method," in *Proceedings of the Pattern Learning Method Asia-Pacific Materials Science and Information Technology Conference (APMSIT '14)*, Shanghai, China, December 2014.

[19] Q.-C. Bui, P. M. A. Sloot, E. M. Van Mulligen, and J. A. Kors, "A novel feature-based approach to extract drug-drug interactions from biomedical text," *Bioinformatics*, vol. 30, no. 23, pp. 3365–3371, 2014.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," 2012.

[22] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[23] A. Fischer and C. Igel, "Progress in pattern recognition, image analysis, computer vision, and applications," in *17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3–6, 2012. Proceedings*, An Introduction to Restricted Boltzmann Machines, pp. 14–36, Springer, Berlin, Germany, 2012.

[24] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*, pp. 1064–1071, 2008.

[25] G. E. Dahl, R. P. Adams, and H. Larochelle, "Training restricted boltzmann machines on word observations," https://arxiv.org/abs/1202.5695.

[26] Y. Tang and I. Sutskever, *Data Normalization in the Learning of Restricted Boltzmann Machines*, Department of Computer

Science, Toronto University UTML-TR-11, Toronto, Canada, 2011, http://www.cs.toronto.edu/~tang/papers/RbmZM.pdf.

[27] G. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., vol. 7700 of *Lecture Notes in Computer Science*, pp. 599–619, Springer, Berlin, Germany, 2nd edition, 2012.

[28] H. Chen and A. F. Murray, "Continuous restricted Boltzmann machine with an implementable training algorithm," *IEE Proceedings—Vision, Image and Signal Processing*, vol. 150, no. 3, pp. 153–158, 2003.

[29] M. Welling, M. Rosen-Zvi, and G. E. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Advances in Neural Information Processing Systems (NIPS) 17*, pp. 1481–1488, 2005.

[30] A. Ng, Deep Learning Tutorial, http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] J. Hammerton, "Named entity recognition with long short-term memory," in *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL (CONLL '03)*, vol. 4, pp. 172–175, 2003.

[33] C. Olah, "Understanding LSTM Networks," 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[34] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 3, p. 6, Como, Italy, July 2000.

[35] S. Hochreiter, "LSTM Can Solve Hard," in *Advances in Neural Information Processing Systems*, Neural Information Processing Systems (NIPS), Denver, Colo, USA, 1996.

[36] S. Otte, D. Krechel, and M. Liwicki, "Jannlab neural network framework for java," in *Proceedings of the Poster Proceedings Conference (MLDM '13)*, pp. 39–46, IBAI, New York, NY, USA, 2013.

[37] R. Boyce and G. Gardner, "Using natural language processing to identify pharmacokinetic drug-drug interactions described in drug package inserts," in *Proceedings of the Workshop on Biomedical Natural Language Processing (BioNLP '12)*, pp. 206–213, BioNLP, Montreal, Canada, 2012.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 7942501, 12 pages
http://dx.doi.org/10.1155/2016/7942501

Research Article

Objects Classification by Learning-Based Visual Saliency Model and Convolutional Neural Network

Na Li,[1] Xinbo Zhao,[1] Yongjia Yang,[1] and Xiaochun Zou[2]

[1]*School of Computer Science, Northwestern Polytechnical University, Xi'an, China*
[2]*School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China*

Correspondence should be addressed to Xinbo Zhao; xbozhao@nwpu.edu.cn

Received 27 May 2016; Revised 30 July 2016; Accepted 24 August 2016

Academic Editor: Trong H. Duong

Copyright © 2016 Na Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Humans can easily classify different kinds of objects whereas it is quite difficult for computers. As a hot and difficult problem, objects classification has been receiving extensive interests with broad prospects. Inspired by neuroscience, deep learning concept is proposed. Convolutional neural network (CNN) as one of the methods of deep learning can be used to solve classification problem. But most of deep learning methods, including CNN, all ignore the human visual information processing mechanism when a person is classifying objects. Therefore, in this paper, inspiring the completed processing that humans classify different kinds of objects, we bring forth a new classification method which combines visual attention model and CNN. Firstly, we use the visual attention model to simulate the processing of human visual selection mechanism. Secondly, we use CNN to simulate the processing of how humans select features and extract the local features of those selected areas. Finally, not only does our classification method depend on those local features, but also it adds the human semantic features to classify objects. Our classification method has apparently advantages in biology. Experimental results demonstrated that our method made the efficiency of classification improve significantly.

1. Introduction

Objects classification is one of the most essential problems in computer vision. It is the basis of many other complex vision problems, such as segmentation, tracking, and action analysis. And objects classification has wide application in many fields, such as security, transportation, and medicine. Thus, computer automatic classification technology can lighten the burden of people and change people's life style.

Humans have the powerful ability of visual perception and objects classification. When they classify different objects, they firstly select information by visual pathway, and then their nervous system make correct decision without needing extensive training by using this selected information (Figure 1). If computer can mimic the ability of humans, computer automatic classification technology will be improved greatly. To achieve this assumption, we combine simulation of human visual information processing mechanism and simulation of human neutral network (Figure 2).

Referring to the research results from cognitive psychology and neuroscience, we can build learning-based visual attention model as human visual information processing mechanism. Most models of attention [1–3] are biologically inspired. But some of them are only based on a bottom-up computational model, which does not match the human behavior. Other models of attention such as Cerf et al. [4] combine low-level visual features and high-level visual features, but most of them were under the "free-viewing"; it cannot be used to analyze and predict the region of interest when people classify different objects. To address this problem, we build a task-based and learning-based visual attention model combining low-level and high-level image features to obtain the humans' classification RoI (region of interest).

Deep learning is good to disentangle abstractions and pick out which features are useful for learning like human brain does, so we can use deep learning method to simulate the human neutral network. Convolutional neural network (CNN) as one of methods of deep learning can be used to solve classification problem. CNN was inspired by biological processes [5], which is a type of feed-forward artificial neural network. It is inspired by the organization of the animal visual

Figure 1: The picturesque processing of humans classifying different objects. Person firstly selects information by visual pathway, and then his nervous system uses this selected information to make correct decision without needing extensive training [6].

cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field. Compared to other image classification algorithms, CNN uses relatively little preprocessing. The lack of dependence on prior knowledge and human effort in designing features is a major advantage of CNN, which lead CNN to be more suitable for solving computer automatic classification problem.

In this paper, we make five contributions. Firstly, for learning common people visual behaviors when they classify different objects, we established an eye-tracking database and recorded eye-tracking data of 10 viewers on 300 images. Secondly, to simulate human visual information processing mechanism when they were asked to classify different objects, we used EDOC database as training and testing examples to learn a learning-based visual attention model based on low-level and high-level image features and then analyzed and predicted the humans' classification RoIs. Next, seeing that the CNN is inspired by biological processes and has remarkable advantages, we established a CNN framework to simulate the human brain's processing of classification. But, unlike traditional CNN, we use RoIs predicted from our learning-based visual attention model as the input of CNN, and thus it will be more close to human. Furthermore, for improving the biological advantages of our computer automatic classification method, we combine the high-level features also used in our visual attention model with local features gained by our CNN network to classify objects by SVM. Finally, we established big database ImageSix, including 6000 images to testify the robustness of our classification method.

And all experimental results showed that our method made the efficiency of classification improve significantly.

2. Related Work

Objects classification is one of hot problems in computer vision. Humans recognize a multitude of objects in images with little effort; however, this task is still a challenge for computer vision systems. Many approaches to the task have been implemented over many decades, such as approaches based on CAD-like object models [7], appearance-based methods [8–11], feature-based methods [12–14], and genetic algorithm [15]. These traditional approaches perform well in some fields, but they are not suitable for multiple-classes objects classification. Currently, the best algorithms for this problem are based on convolutional neural networks. An illustration of their capabilities is given by the ImageNet Large Scale Visual Recognition Challenge; this is a milestone in object classification and detection, with millions of images and hundreds of object classes. And performance of convolutional neural networks on the ImageNet tests is now close to that of humans.

For being more close to humans, it is very significant to bring visual saliency model to CNN as our method, because common CNN ignores the idea that human visual system has a major part to select information before classification. So we develop a learning-based visual attention model.

In the past few years, there were many researches on human eye movements, and many saliency models based on various techniques with compelling performance exist, but most of them were under the "free-viewing." One of the most influential ones is a pure bottom-up attention model proposed by Itti et al. [16], based on the feature integration theory [17]. In this theory, an image is decomposed into low-level attributes such as color, orientation, and intensity. Based on the idea of decorrelation of neural responses, Garcia-Diaz et al. [18] proposed an effective model of saliency known as Adaptive Whitening Saliency (AWS). Another class of models is based on probabilistic formulation. Zhang et al. [19] put forward SUN (Saliency Using Natural statistics) model in which bottom-up saliency emerges naturally as the self-information of visual features. Similarly, Torralba [20] proposed a Bayesian framework for visual search which is also applicable for saliency detection. Graph Based Visual Saliency (GBVS) [21] is another method based on graphical models. Machine learning approaches have also been used in modeling visual attention by learning models from recorded eye fixations. For learning saliency, Schölkopf et al. [22] used image patches and Tilke et al. [23] used a vector of several features at each pixel.

These computational models have been used to characterize RoIs in natural images, but their use in classification has remained very limited. But their features extraction method has been proven certainly effective. When we built a visual attention model for classification problem, we learnt these computational models' features extraction method for guidance.

3. Learning a Saliency Model for Objects Classification

3.1. Database of Eye-Tracking Data. For learning common people visual behaviors when they classify different objects and recording their eye-tracking data, we established an eye-tracking database, including six kinds of objects such as aeroplanes, bikes, cars, dogs, persons, and white cats, called EDOC database (eye-tracking database for objects classification) (Figure 3). The EDOC allows quantitative analysis of fixation points and provides ground truth data for saliency model research as well as labels for each class. Compared with several eye-tracking datasets that are publicly

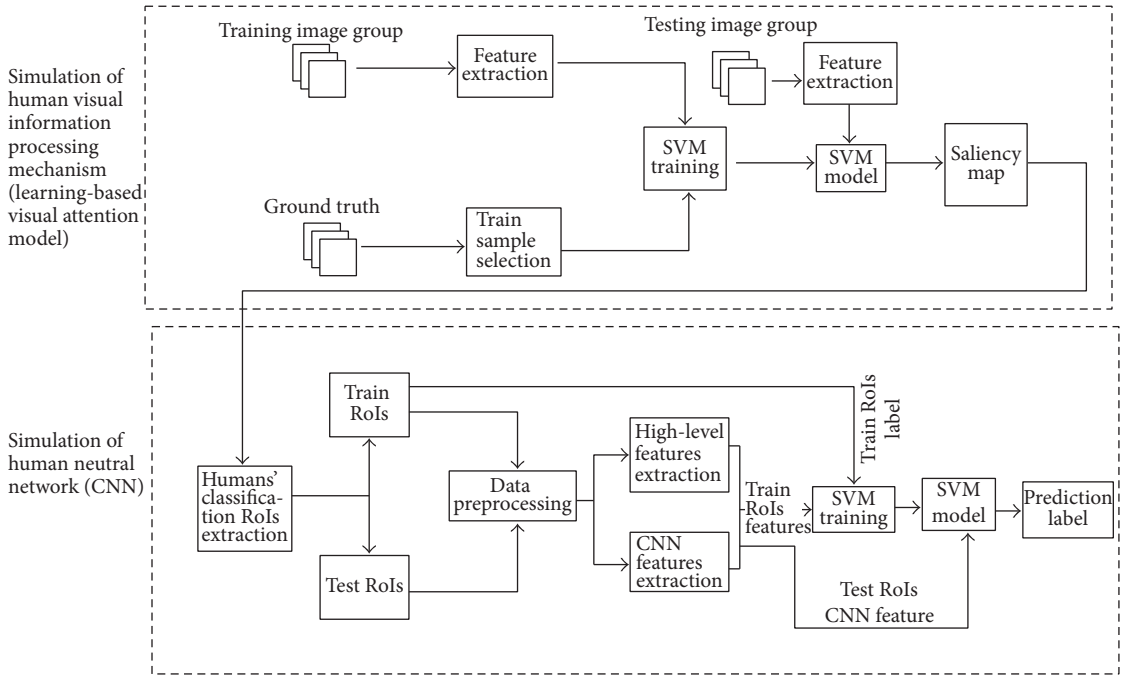


Figure 2: The algorithm flow chart of this paper. We establish a learning-based visual saliency model to simulate human visual information processing mechanism and then obtain saliency map which can be used to get the humans' classification RoIs. CNN is used to simulate human neutral network, and the humans' classification RoIs is CNN's input. After the processing of CNN, we obtain the result of classification which is close to humans.

available, the main motivation of our new dataset is for objects classification.

The purpose of the current analysis was to model the classification process of visual selection of relevant regions in different objects images. We collected 50 images for each class of objects and 300 images (Figure 4(a)) altogether, which are stored in JPEG format. And we recorded eye-tracking data from ten subjects, including 5 females and 5 males, whose age range from 12 to 40. Subjects were asked to view these images to find the most representative regions of each class (Figure 4(b)), which can be used to differentiate the six classes objects.

We used a Tobii TX300 Eye Tracker device to record eye movements, which is at a sample rate of unique combination of 300 Hz. The TX300 Eye Tracker device has very high precision and accuracy and robust eye tracking; besides, it also has compensation for large head movements extending the possibilities for unobtrusive research of oculomotor functions and human behavior. Although it has a variety of researcher profiles, subjects can use the system without needing extensive training.

In the experiments, each image was presented for 5 s followed by a rapid and automatic calibration procedure. To ensure high-quality tracking results, we checked camera calibration every 10 images. During the first 1 s viewing, subjects maybe free viewed the images, so we discarded the first 1 s viewing tracking results of each subject. In order to obtain a continuous ground truth of an image from the eye-tracking data of a subject, we convolved a Gaussian filter across the subject's fixation locations, similar to the "landscape map." We overlapped the eye-tracking data collected from all subjects (Figure 4(c)) and then generated ground truth of the average locations (Figure 4(d)).

3.2. Learning-Based Visual Attention Model. In contrast to manually designed measures of saliency, we follow a learning approach by using statistics and machine learning methods directly from eye-tracking data (Figure 2, simulation of human visual information processing mechanism). As shown in Figure 5, a set of low-level visual features are extracted from some training images. After the feature extraction process, the features of the top 5% (bottom 30%) points in the ground truth are selected as training samples in each training image. All of the training samples are sent to train a SVM model. Then, a test image can be decomposed into several feature maps and imported into SVM model to predict the saliency map. After the saliency map prediction, we can use them to obtain the human classification RoIs as inputs of CNN to continue solving classification problem.

After analyzing the EDOC dataset, we first extract a set of features for every pixel in each $m \times n$ pixels image. We recomputed the 35 features including 31 low-level features and 4 high-level features, for every pixel of the each image which resized to 200×200, and used these to train our visual attention model (Ours). The following are the low- level and high-level features (Figure 5) with which we were motivated to work after analyzing our dataset (Figure 2, simulation of human neural network).

(1) Low-Level Features. Because of the underlying biological plausibility [17], low-level features have been shown to correlate with visual attention. We use 31 low-level features:

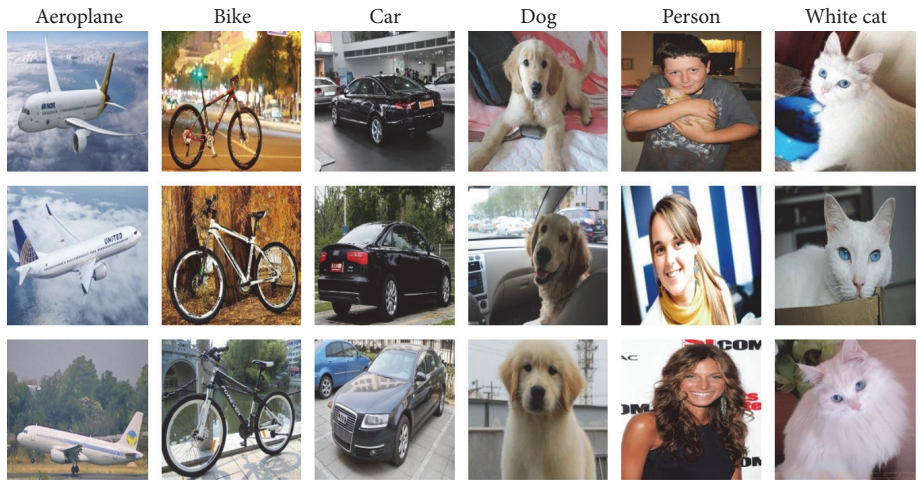
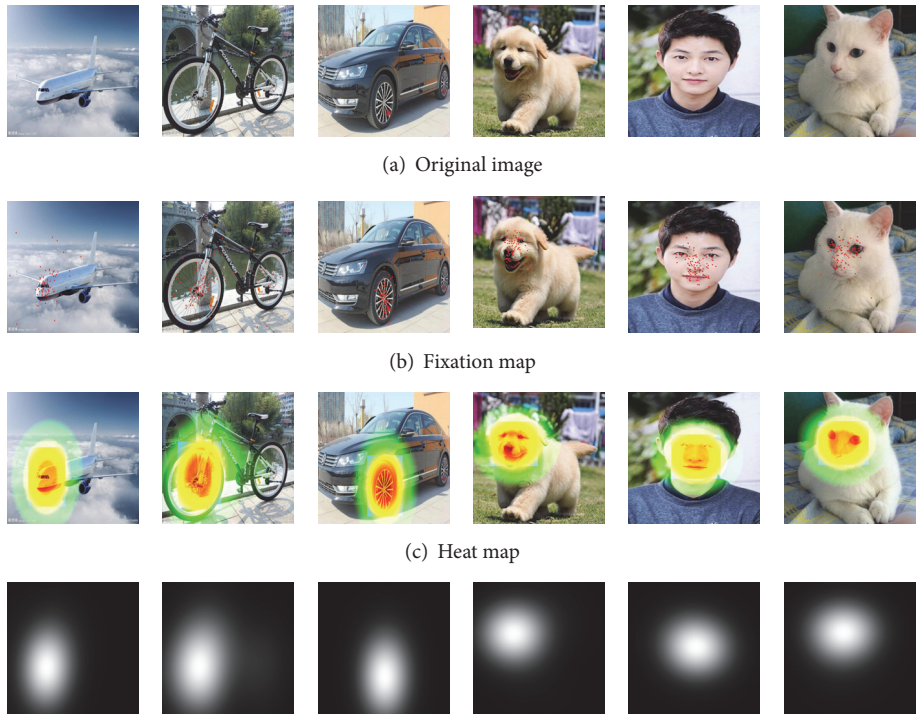


FIGURE 3: Images. A sample of the 300 images of EDOC. Though they were shown at original resolution and aspect ratio in the experiment, they have been resized for viewing here.



(a) Original image

(b) Fixation map

(c) Heat map

(d) Ground truth

FIGURE 4: We collected eye-tracking data on 300 images from ten subjects. The first row is the sample of original images (a) in EDOC. Gaze tracking paths and fixation locations are recorded in the second row (b). The third row (c) heat maps show RoIs according to (b). A continuous ground truth (d) is found by convolving Gaussian over the fixation locations of all subjects.

(a) The local energy of the steerable pyramid filters [24] is used as features in four orientations and three scales (Figure 5, the first 13 images).

(b) We include intensity, orientation, and color contrast corresponding to image features as calculated by Itti and Koch's saliency [2] (Figure 5, images 14 to 16), because the three channels have long been seen as important features for bottom-up saliency.

(c) We include features used in a simple saliency model described by Torralba [25] and GBVS [21] and AWS [26] based on subband pyramids (Figure 5, images 17 to 19).

(d) The values of the red, green, and blue channels, as well as the probabilities of each of these channels, are used as features (Figure 5, images 20 to 25) in addition to the probability of each color as computed from 3D

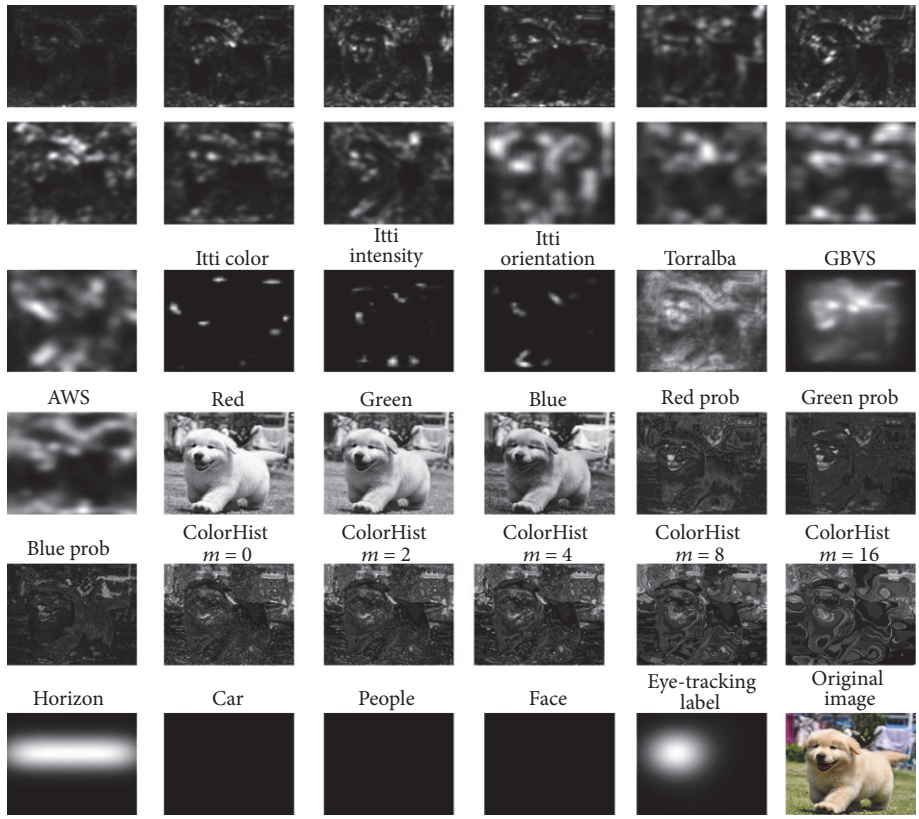


Figure 5: Features. A sample image (bottom right) and 35 of the features that we use to train the model. These include subband features, Itti and Koch saliency channels, three simple saliency models described by Torralba and GBVS and AWS, color features and automatic horizon, car, people, and face detectors. The labels for our training on this image are based on a threshold saliency map derived from human fixations (to the left of bottom right).

color histograms of the image filtered with a median filter at six different scales (Figure 5, images 26 to 30).

(e) The horizon is a place where humans naturally look for salient objects, because most objects rest on the surface of the earth. So we use the horizon as the last low-level feature (Figure 5, images 31).

(2) High-Level Features. In the light of the eye-tracking data obtained from our experiment, we found that humans fixated so consistently on people, faces, and cars, so we run the Viola Jones face detector [27] and the Felzenszwalb person and car detector [28] and include these as features to our model (Figure 5, images 32 to 35).

4. CNN for Feature Extraction

Convolutional neural network (CNN) was initially proposed by Cun et al. in the early 1980s [29]. Following the discovery of human visual mechanisms, local visual field is designed to make the CNN deep and robust in the 1990s. CNN is a neural network model, whose weight sharing network makes itself more similar to biological neural network, reducing the complexity of network model and the number of weight. CNN is based on four key architectural ideas: local receptive fields, convolution, weight sharing, and subsampling in the spatial domain. A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume through a differentiable function. In a CNN structure, convolutional layers and subsampling layers are connected one by one and trained by supervised learning method with labeled data, the architecture of the CNN we used is shown in Figure 6, and the labeled data we used to train the CNN is obtained from our visual attention model. Due to the neuron network simulation, the CNN is usually used as a strong feature extractor and has achieved great success on image processing fields.

4.1. Convolution Layers. The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of kernels, which have a small receptive field, but extend through the full depth of the input volume. At a convolution layer, the previous layer's feature maps are convolved with learnable kernels and put through the activation function to form the output feature map. Each output map may combine convolutions with multiple input maps.

By training, kernels can extract several meaningful features; for example, the first convolutional layer is similar to Gabor filter, which can extract the information of corner, angle, and so forth. The CNN we used contains 4 convolutional layers (C1~C4), the kernel size, respectively, is 5, 5, 5, and 4 pixels, the number of feature maps, respectively, is 9, 18, 36, and 72, and all of the stride is 1 (Figure 6). Multilayers

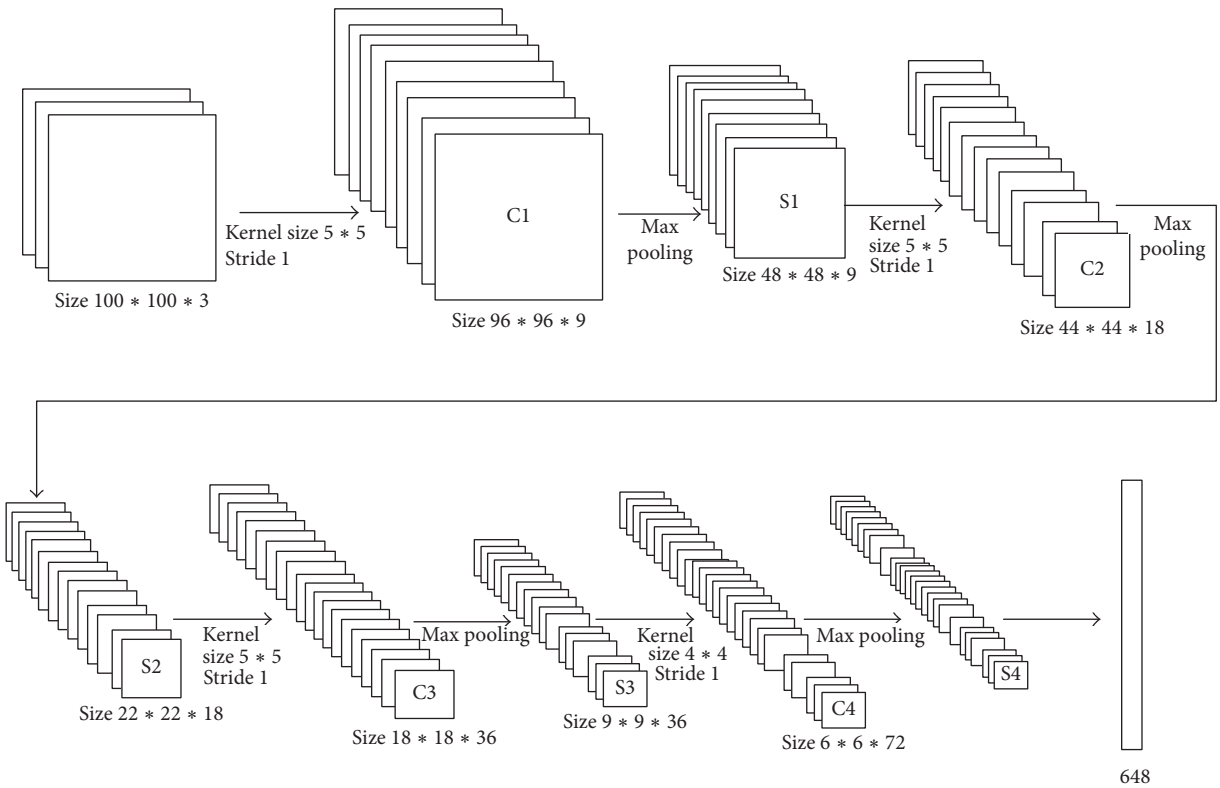


Figure 6: An illustration of the architecture of our CNN. The CNN we used contains 4 convolutional layers (C1~C4), the kernel sizes, respectively, are 5, 5, 5, and 4 pixels, the number of feature maps, respectively, is 9, 18, 36, and 72, and all of the stride is 1. All of the subsampling (S1~S2) size, respectively, is 2 pixels, and all of the stride is 1. The network's input is 3000 dimension features and output is 648 dimension features.

structure can abstract the input image layer by layer, to obtain a higher level distributed feature expression.

4.2. Subsampling Layers. Another important concept of CNNs is subsampling, which is a form of nonlinear down-sampling. There are several nonlinear functions to implement subsampling among which max pooling is the most common. It partitions the input image into a set of nonoverlapping rectangles and, for each such subregion, outputs the maximum. The intuition is that once a feature has been found, its exact location is not as important as its rough location relative to other features.

A subsampling layer produces downsampled versions of the input maps. If there are N input maps, then there will be exactly N output maps, although the output maps will be smaller.

The CNN we used contains 4 subsampling layers (S1~S4), which are periodically inserted in between successive convolutional layers. All of the subsampling size, respectively, is 2 pixels, and all of the stride is 1 (Figure 6). Multilayers structure can abstract the input image layer by layer, to obtain a higher level distributed feature expression. By subsampling, we can not only reduce the dimension of features, but also improve their robustness.

4.3. Parameter Sharing. Parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on one reasonable assumption; that is, if one patch feature is useful to compute at some spatial position, then it should also be useful to compute at a different position.

Since all neurons in a single depth slice are sharing the same parametrization, then the forward pass in convolutional layer can be computed as a convolution of the neuron's weights with the input volume. Therefore, it is common to refer to the sets of weights as a kernel, which is convolved with the input. Parameter sharing contributes to the translation invariance of the CNN architecture.

4.4. Fully Connected Layer. Finally, after several convolutional and max subsampling layers, the high-level reasoning in the neural network is done via fully connected layers and the CNN we used contains one fully connected layer. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with matrix multiplication followed by a bias offset.

So far, the structure of our CNN network contains four convolutional layers, four subsampling layers, and one fully connected layer. We use the humans' classification RoIs obtained from visual attention model as the input of our CNN network, after feature extracting. Our CNN network outputs 648 dimension local features, which are parts of features used to classify objects.

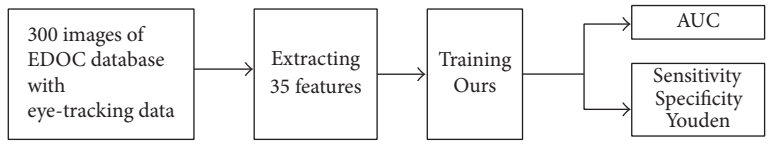


Figure 7: The whole processing of evaluating our visual attention model. We trained Ours after extracting features in EDOC database and measure Ours by AUC, sensitivity, specificity, and Youden and compared it with other eight visual attention models.

5. Objects Classification

In order to be more close to humans' classification behavior, we build a task-based and learning-based visual attention model which combines low-level and high-level image features to obtain the humans' classification RoIs. Then, we construct CNN network to extract more features of those humans' classification RoIs. Although CNN is based on the neuron network simulation and is a strong feature extractor, the features obtained by CNN are the group of local features. However, humans always analyze images by putting them into context. Thus, for improving the biological advantages of our computer automatic classification method, we combine the 3 dimension high-level features also used in our visual attention model, including people, faces, and cars, with 648 dimension local features gained by our CNN network to classify objects.

Developing from statistics, the theory of SVM is a general learning method, which has excellent generalization ability in nonlinear classification, function approximation, and pattern recognition. Even though the sample is limited, SVM can effectively construct high-dimensional data model, can converge to the global optimum, and is insensitive to dimensions. Owing to the advantages of SVM, we use it to classify objects after acquiring 651 dimension features. The detailed processing of our classification method is shown in Figure 2.

6. Experimental Result and Discuss

In order to validate our classification method, we perform four experiments. (1) Section 6.1 evaluates our visual attention model (Ours) and compares it with other eight visual attention models. (2) Section 6.2 compares the classification results of using humans' classification RoIs as input of classification and using original images as input of classification. (3) Section 6.3 compares the classification results when only using features extracted by CNN and when combining high-level features and local features extracted by CNN. (4) Section 6.4 validates our classification method in 6000 images. In Sections 6.2, 6.3, and 6.4, we all use the error rate of classification and convergence rate as evaluation criterion. And our experiments were all based on a server IBM x3650m5, with CPU E5-2603v2 (2.4 GHz) and 32 GB RAM.

6.1. Performance of Our Visual Attention Model. We validate our visual attention model by applying it to humans' classification RoIs prediction; the whole processing of this experiment is shown in Figure 7. We used EDOC database to evaluate our results; images were resized in 200 × 200 pixels. We randomly used 30 images of each class as training data and 20 images of each class as testing data. The database provided subjects' eye-tracking data as ground truth.

Since there is no consensus over a unique score for saliency model evaluation, a model that performs well should have good overall scores. We measure performance of saliency models in the following two ways.

First, we measure performance of each model by Area Under the ROC Curve (AUC). AUC is the most widely used metric for evaluating visual saliency. When AUC is equal to 1, the two distributions are exactly equal, not relative when AUC is equal to 0.5, and negatively relative when AUC is equal to 0.

Second, three quality measurements, classical sensitivity, specificity, and Youden, were computed. Sensitivity, also called the true positive rate, measures the proportion of positives which are correctly identified and is complementary to the false negative rate. The higher the sensitivity is, the more sensitive the test is. Specificity, also called the true negative rate, measures the proportion of negatives which are correctly identified and is complementary to the false positive rate. The higher the specificity is, the more precise the test is. Youden, called Youden, can be written as formula (1), whose value ranges from 0 to 1. The higher Youden is, the higher authenticity the test has. Besides, Youden gives equal weight to false positive and false negative values. Consider

$$\text{Youden} = \text{sensitivity} + \text{specificity} - 1. \quad (1)$$

6.1.1. Analysis of AUC. Our method is biologically inspired. The developed method was compared with eight well-known techniques which dealt with similar challenges. These eight models were AIM [30], AWS [26], Judd [23], Itti [16], GBVS [21], SUN [19], STB [26], and Torralba [31]. We used them as the baseline because they also mimic the visual system. In the experiment, we randomly chose 30 images over the dataset of each class to train our model and the rest 20 images were used for testing. The statistical results are shown in Table 1.

Table 1 shows the comparison of evaluation performances of the 9 models in the EDOC database. In this experiment, the average values of six classes in Table 1 are used for comparison. In the results, Ours has the best value in AUC. The AUC of our model is highest (0.8421), followed by Judd (0.8287) and GBVS (0.8284). However, the average is only 0.7642. It means the results of Ours are more identical with ground truth than other models. Generally speaking, Ours has good performance in this metric. And Figure 8 presents

Table 1: Performance comparison of nine models in the EDOC dataset.

Metrics	GT	Ours	AIM	AWS	GBVS	ITTI	STB	SUN	Torralba	Judd	Average
AUC	1.0000	0.8421	0.7232	0.7811	0.8284	0.6078	0.8151	0.7360	0.7158	0.8287	0.7642

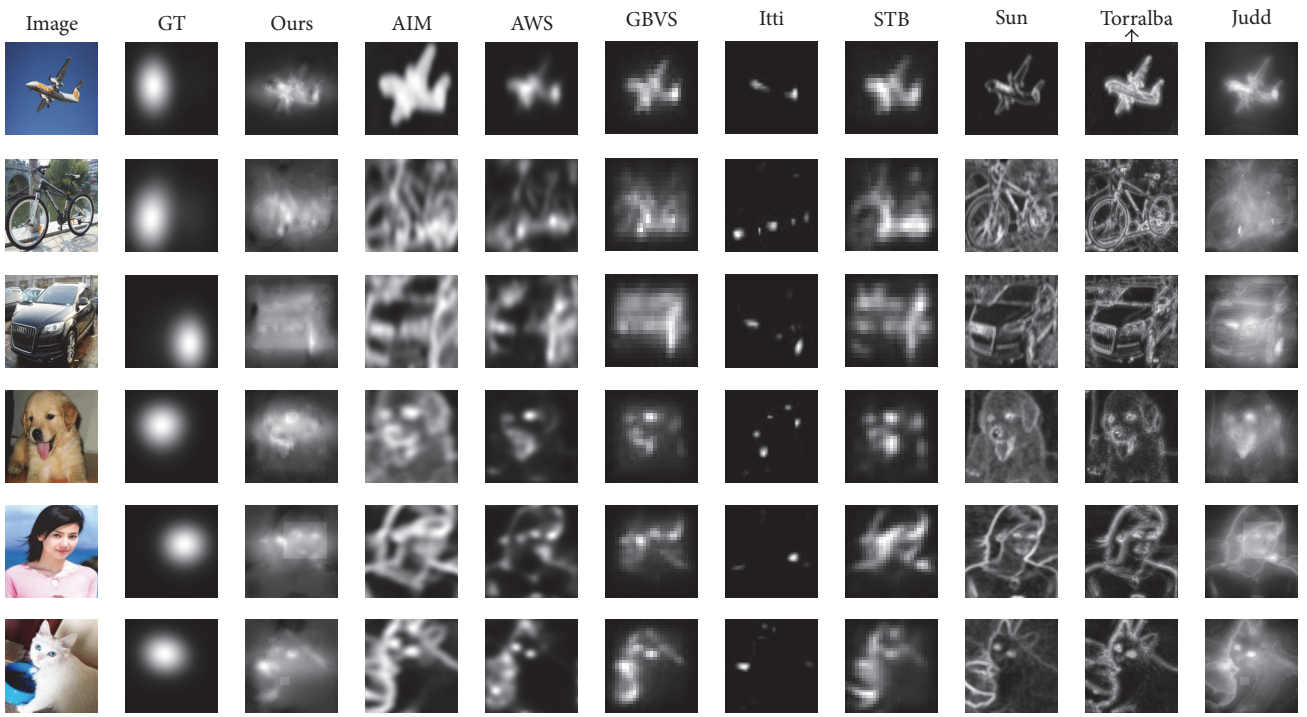


Figure 8: Some saliency maps are produced by 9 different models from the EDMERI database along with predictions of several models using ROC. Each example is shown by one row. From left to right: original image, ground truth, Ours, AIM, AWS, GBVS, Itti, STB, SUN, Torralba, and Judd. It is obvious that Ours is more similar to the ground truth than other saliency maps.

six examples of the saliency maps produced by our approach and the other eight saliency models.

6.1.2. Analysis of Sensitivity, Specificity, and Youden. The ability of the different methods to predict humans' classification visual saliency maps was evaluated using conventional sensitivity, specificity, and Youden measurements. These results are shown in Table 2.

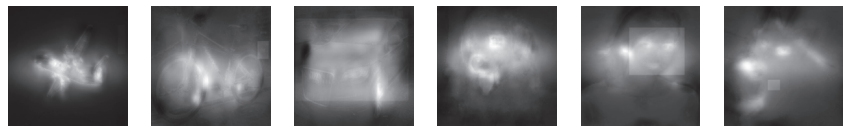
Table 2 shows sensitivity and specificity and Youden of the 9 models in 60% salient region. Overall, all sensitivity, specificity, and Youden measurements evidence that our model outperforms the other models. The sensitivity of our model is 73.2895%, which surpasses the average sensitivity 13.1638%, followed by Judd with 71.9905% and GBVS with 71.4794%. However, Itti had the lowest rate (only 40.8605%), less than approximately half of Ours. And the larger value of specificity (82.2354%) is also shown in our model, which exceeds the average specificity 4.0582%. Besides, the sensitivity of other models are all under 80% and under Ours. Owing to having the highest value of sensitivity and specificity, Youden (0.5552) of our model is the highest among the 9 models, followed by Judd with 0.967 and GBVS with 0.4934. Average Youden is 0.3830, which is only higher than half of Ours. The indisputable fact is that the higher Youden is, the higher authenticity the test has, and our model outperforms the other models in all sensitivity, specificity, and Youden measurements based on Table 2. Thus, Ours is suitable for predicting humans' classification visual saliency maps from images.

6.2. Comparison of Humans' Classification RoIs and Original Images. To testify that humans' classification RoIs outperform original images in objects classification, we used humans' classification RoIs (Figure 9) obtained by the original images of EDOC database and humans' classification visual saliency maps to classify objects and then compare the result of classification with outcome of the experiment when using the original images of EDOC database as input of classification. All images were resized in 100×100 pixels. We input two groups of images, the original images and humans' classification RoIs to our CNN framework to extract features. As introduced above, the architecture of the CNN we used contained 4 convolutional layers and 4 subsampling layers. For two groups of images, we input them to CNN by 3 times, and the frequency of training of CNN is, respectively, 500, 1000, and 1500. We randomly used 30 images of each class as training data and 20 images of each class as testing data. Finally, we used SVM to classify objects and used the error rate to check out whether using humans' classification RoIs can make the classification results better and the whole

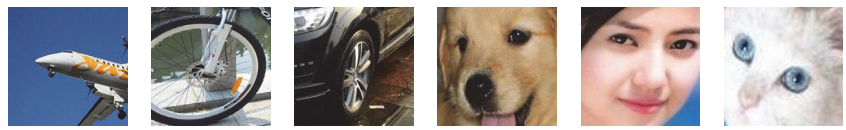
Table 2: Sensitivity, Specificity, and Youden of nine models.

Metrics	Ours	AIM	AWS	GBVS	ITTI	STB	SUN	Torralba	Judd	Average
Sensitivity (%)	73.2895	51.8212	61.1878	71.4794	40.8605	67.9153	49.8252	52.7619	71.9905	60.1257
Specificity (%)	82.2354	77.4174	79.1480	77.8588	77.8217	78.8674	76.0314	76.5402	77.6745	78.1772
Youden	0.5552	0.2923	0.4034	0.4934	0.1868	0.4678	0.2586	0.2930	0.4967	0.3830

(a) Original images



(b) Our saliency maps



(c) Humans' classification RoIs extraction

Figure 9: A sample of input of CNN. (a) is the original images of EDOC database. (b) is saliency maps acquired by our learning-based saliency model. We use (a) and (b) that can extract the humans' classification RoIs (c) which are the input of our CNN framework.

Table 3: The error rate of the classification results by three different frequency trainings of CNN in two groups of input images.

Input images	Frequency of training	500	1000	1500
300 original images	Error rate%	73.3	50.0	36.5
Humans' classification RoIs		63.3	36.7	18.2

Table 4: The comparison of error rate of the classification's results in two features extracting ways.

Features	Frequency of training	500	1000	1500
Features extracting by CNN	Error rate%	63.3	36.7	18.2
High-level features and features extracting by CNN		51.7	25.8	14.2

processing of this experiment as Figure 10 showed. The error rates of the classification results by three different frequency trainings, 500, 1000, and 1500 in two groups of input images, including original images and humans' classification RoI, are shown in Table 3.

Table 3 demonstrates the error rate of the experiments' result by three different frequency trainings in two groups of input images. Overall, all results evidence that our method based on humans' classification RoIs exceeds traditional CNN based on original images. Although when the frequency is 500, the error rate of two method is all more than 50%, our method is less than traditional method 10%. With the increasing of the frequency of training, the error rate of our method based on humans' classification RoIs drops quickly from 63.3% to 18.2%. However, the error rate of traditional CNN based on original images is 50% when the frequency of training is 1000, and even when the frequency of training is 1500, it is still more than 30%. Besides, along with increasing of frequency of training, the error rate will be lower. As we all know, the error rate is lower, and the results of classification are better. So it is not denied that humans' classification RoIs can make the results of classification better.

6.3. Combining High-Level Features and Features Extracted CNN. To prove that combining high-level features and local features extracted by CNN can make the results of classification better, we performed an experiment which added high-level features to SVM model to classify objects and then compared the classification's result of experiment 6.2. And the other settings of experiment 4 were the same as experiment 3 and the whole processing of this experiment as Figure 11 showed. And the comparison of error rate of the classification's results in two features extracting ways is shown in Table 4.

Table 4 shows the comparison of error rate of the classification's results in two features extracting ways. On balance, all results evidence that the classification way based on combining two types of features exceeds only based on features extracting by CNN. When the frequency is 500, the error rate of comprehensive method is 51.7%, which is less

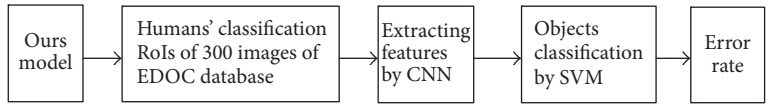


Figure 10: The whole processing of comparing the classification results when using humans' classification RoIs as input of classification and when using original images as input of classification. Different from traditional classification method using original images for classification, our classification method uses humans' classification RoIs as input of classification. After extracting features by CNN, we use SVM model to classify objects and then use the error rate of classification and convergence rate as evaluation criterion.

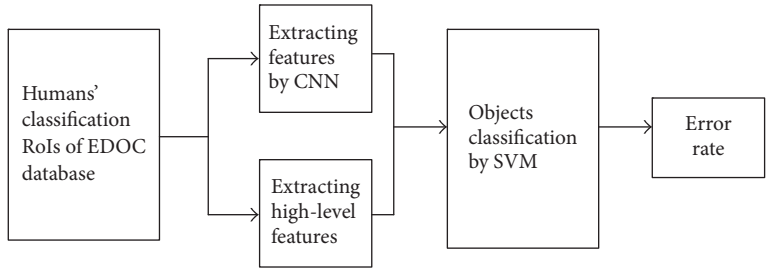


Figure 11: The whole processing of combining high-level features and features obtained by CNN to classify objects. Before SVM model, we added high-level features and then use the error rate of classification and convergence rate as evaluation criterion.

than the single method nearly 12%. According to Table 4, we can conclude that the bigger frequency of training is, the lower error rate will be. But when the frequency of training is 1000, the error rate of comprehensive method (25.8%) is also less than the single method (36.7%) nearly 10%. Most of all, when the frequency of training is 1500, the error rate of comprehensive method (14.2%), which is less than the single method 4%, is almost half of the error rate of the method (36.5%) using original images according to the Table 3. Hence, adding high-level features can make the classification results better.

6.4. Performance of Our Classification Method in 6000 Images Classification. Sections 6.1, 6.2, and 6.3 are all based on the 300 images of EDOC database; the numbers of images are not big, but there is not suitable and available big database including the six classes objects to testify the robustness of our classification method. Thus, we construct big database ImageSix (Figure 12), including 6000 images from the Internet. Firstly, we used Ours to predict the humans' classification RoIs of the images of ImageSix database. Secondly, we extracted the local features of these humans' classification RoIs by CNN. Then, we combined these local features with high-level features extracted by Ours to perform three classification experiments by SVM, and the frequency of training of CNN was also, respectively, 500, 1000, and 1500. For SVM, we randomly used 600 images of each class as training data and 400 images of each class as testing data. The whole processing of this experiment is shown in Figure 13. Finally, we compared the classification's results of our method with outcome of classification method which used the original images as input and extracted features only based on CNN, and the experiment's results are shown in Table 5.

Table 5 shows the error rate of the experiments' result by two methods in ImageSix database. From it, we can conclude that, with the increasing of the number of the training images, the error rate of two methods both drop, but all results evidence that our classification method exceeds the classification method without improvement. When the frequency is 500, the error rate of our method is 44.6%, which is less than classification method without improvement (63.2%) nearly 20% and is even less than the error rate of classification method without improvement (56.7%) in the 1000 frequencies of training. With increasing of frequency of training, the error rate will be lower. However, when the frequency of training is 1500, the error rate of classification method without improvement is still more than 47%. With the increasing of the frequency of training, the error rate of our method drops quickly from 44.6% to 29.1%. Thus, our classification method can make the classification results better without doubt.

Table 5: The error rate of the classification results by three different frequency trainings of CNN in two groups of input images.

Method	Frequency of training	500	1000	1500
Without improvement	Error rate%	63.2	56.7	46.9
Our classification method		44.6	33.8	29.1

7. Conclusion and Discussion

The present paper has introduced a new classification method which combines learning-based visual saliency model and CNN. This method inspired the completed processing that humans classify different kinds of objects and has apparently advantages in biology.

Firstly, we established a database, called EDOC, to learn common people visual behaviors and record their eye-tracking data when they classify different objects.

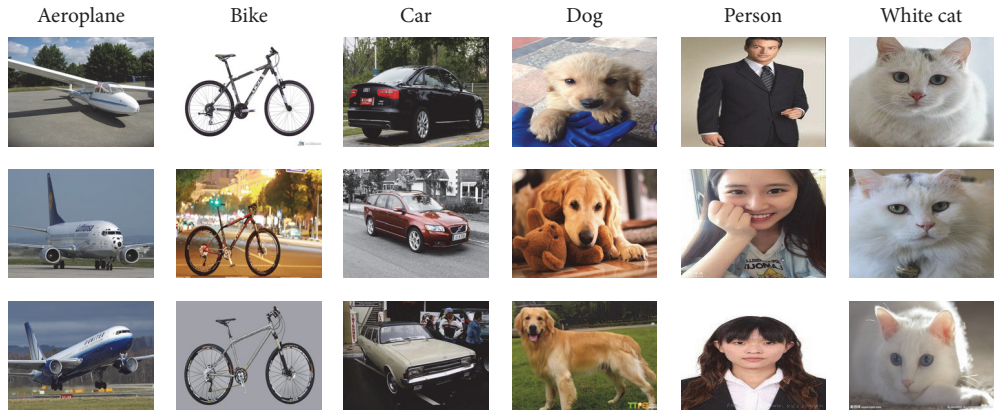


Figure 12: Images. A sample of the 6000 images of ImageSix database.

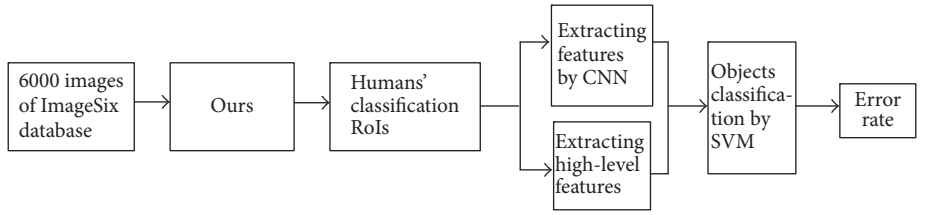


Figure 13: The whole processing of our classification method validated by ImageSix database. First, we predicted humans' classification RoIs of original images in ImageSix database. Second, we combined the features extracted by CNN and high-level features to classify objects by SVM and then used the error rate of classification as evaluation criterion.

Secondly, we built a learning-based visual saliency model trained by EDOC database. Our model has the ability to automatically learn the relationship between saliency and features. And our model simultaneously considers appearing frequency of features and the pixel location of features, which intuitively have a strong influence on saliency. As a result, our model can determine saliency regions and predict humans' classification RoIs more precisely.

Then, we built a CNN framework and used humans' classification RoIs obtained from our visual attention model to train CNN; thus, it will be closer to humans.

Finally, for improving the biological advantages of our computer automatic classification method, we combined the 3 dimension high-level features with 648 dimension local features gained by our CNN network to classify objects by SVM.

To evaluate every aspect of our classification method, we performed 4 groups of experiments. In particular, we established a big ImageSix database, including 6000 images to testify the robustness of our classification method. And all experimental results showed that our method made the efficiency of classification improve significantly.

Our classification method is inspired by the completed processing that humans classify different kinds of objects; however, it is not denied that human thinking process is so sophisticated; we cannot copy the full processing. Besides, for different objects, human thinking process is quite different. So, in the future, to improve the performance of our method, we can optimize the processing of feature extraction and build different CNN framework for different objects; meanwhile, it will become very costly.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The work is supported by NSF of China (nos. 61117115 and 61201319) and sponsored by "the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University" (Z2016155) and "New Talent and Direction" program.

References

[1] X. Hou and L. Zhang, "Saliency detection: a spectral residual approach," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–8, IEEE Computer Society, Minneapolis, Minn, USA, June 2007.

[2] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, no. 10–12, pp. 1489–1506, 2000.

[3] R. Rosenholtz, "A simple saliency model predicts a number of motion popout phenomena," *Vision Research*, vol. 39, no. 19, pp. 3157–3163, 1999.

[4] M. Cerf, J. Harel, W. Einhäuser, and C. Koch, "Predicting human gaze using low-level saliency combined with face detection," in *Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS '07)*, vol. 20, pp. 241–248, Vancouver, Canada, December 2007.

[5] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection

using a convolutional neural network," *Neural Networks*, vol. 16, no. 5-6, pp. 555–559, 2003.

[6] http://www.zhihu.com/question/21557819.

[7] R. Mohan and R. Nevatia, "Perceptual organization for scene segmentation and description," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 6, pp. 616–635, 1992.

[8] M. J. Swain and D. H. Ballard, "Indexing via color histograms," in *Proceedings of the Proceedings 3rd International Conference on Computer Vision*, pp. 390–393, December 1990.

[9] B. Schiele and J. L. Crowley, "Recognition without correspondence using multidimensional receptive field histograms," *International Journal of Computer Vision*, vol. 36, no. 1, pp. 31–50, 2000.

[10] O. Linde and T. Lindeberg, "Object recognition using composed receptive field histograms of higher dimensionality," in *Proceedings of the International Conference on Pattern Recognition*, vol. 2, pp. 1–4, Cambridge, UK, 2004.

[11] O. Linde and T. Lindeberg, "Composed complex-cue histograms: an investigation of the information content in receptive field based image descriptors for object recognition," *Computer Vision & Image Understanding*, vol. 116, no. 4, pp. 538–560, 2012.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] T. Lindeberg, "Scale invariant feature transform," *Scholarpedia*, vol. 7, no. 5, pp. 2012–2021, 2012.

[14] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: speeded up robust features," *Computer Vision & Image Understanding*, vol. 110, no. 3, pp. 404–417, 2006.

[15] K. Lillywhite, D.-J. Lee, B. Tippetts, and J. Archibald, "A feature construction method for general object recognition," *Pattern Recognition*, vol. 46, no. 12, pp. 3300–3314, 2013.

[16] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[17] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human Neurobiology*, vol. 4, no. 4, pp. 219–227, 1985.

[18] A. Garcia-Diaz, X. R. Fdez-Vidal, X. M. Pardo, and R. Dosil, "Decorrelation and distinctiveness provide with human-like saliency," in *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems (Acivs '09)*, vol. 5807, pp. 343–354, Bordeaux, France, September-October 2009.

[19] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: a bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, no. 7, article 32, 2008.

[20] A. Torralba, "Modeling global scene factors in attention," *Journal of the Optical Society of America A: Optics and Image Science, and Vision*, vol. 20, no. 7, pp. 1407–1418, 2003.

[21] B. Schölkopf, J. Platt, and T. Hofmann, "Graph-based visual saliency," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 19, pp. 545–552, MIT Press, 2010.

[22] B. Schölkopf, J. Platt, and T. Hofmann, "A nonparametric approach to bottom-up visual saliency," in *Proceedings of the 2006 Conference in Advances in Neural Information Processing Systems*, pp. 689–696, Vancouver, Cabnada, December 2006.

[23] J. Tilke, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," in *Proceedings of the 12th International Conference on Computer Vision (ICCV '09)*, vol. 30, pp. 2106–2113, October 2009.

[24] E. P. Simoncelli and W. T. Freeman, "The steerable pyramid: a flexible architecture for multi-scale derivative computation," in *Proceedings of the International Conference on Image Processing*, vol. 3, pp. 444–447, Washington, DC, USA, October 1995.

[25] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[26] A. Garcia-Diaz, X. R. Fdez-Vidal, X. M. Pardo, and R. Dosil, "Decorrelation and distinctiveness provide with human-like saliency," in *Advanced Concepts for Intelligent Vision Systems: 11th International Conference, ACIVS 2009, Bordeaux, France, September 28–October 2, 2009. Proceedings*, vol. 5807 of *Lecture Notes in Computer Science*, pp. 343–354, Springer, Berlin, Germany, 2009.

[27] P. Viola and M. Jones, "Robust real-time object detection," in *Proceedings of the IEEE International Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, July 2001.

[28] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, IEEE, Anchorage, Alaska, USA, June 2008.

[29] Y. Cun, B. Le, J. S. Denker et al., "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, vol. 88, p. 465, Morgan Kaufmann Publishers, 1990.

[30] N. D. B. Bruce and J. K. Tsotsos, "Saliency based on information maximization," *Advances in Neural Information Processing Systems*, vol. 18, no. 3, pp. 298–308, 2005.

[31] D. Walther and C. Koch, "Modeling attention to salient proto-objects," *Neural Networks*, vol. 19, no. 9, pp. 1395–1407, 2006.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 3264587, 9 pages
http://dx.doi.org/10.1155/2016/3264587

Research Article

Social Media Meets Big Urban Data: A Case Study of Urban Waterlogging Analysis

Ningyu Zhang, Huajun Chen, Jiaoyan Chen, and Xi Chen

Computer Science and Technology Institute, Zhejiang University, Hangzhou 310058, China

Correspondence should be addressed to Ningyu Zhang; zhangningyu@zju.edu.cn

Received 10 May 2016; Revised 3 July 2016; Accepted 31 August 2016

Academic Editor: Trong H. Duong

Copyright © 2016 Ningyu Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the design and development of smart cities, opportunities as well as challenges arise at the moment. For this purpose, lots of data need to be obtained. Nevertheless, circumstances vary in different cities due to the variant infrastructures and populations, which leads to the data sparsity. In this paper, we propose a transfer learning method for urban waterlogging disaster analysis, which provides the basis for traffic management agencies to generate proactive traffic operation strategies in order to alleviate congestion. Existing work on urban waterlogging mostly relies on past and current conditions, as well as sensors and cameras, while there may not be a sufficient number of sensors to cover the relevant areas of a city. To this end, it would be helpful if we could transfer waterlogging. We examine whether it is possible to use the copious amounts of information from social media and satellite data to improve urban waterlogging analysis. Moreover, we analyze the correlation between severity, road networks, terrain, and precipitation. Moreover, we use a multiview discriminant transfer learning method to transfer knowledge to small cities. Experimental results involving cities in China and India show that our proposed framework is effective.

1. Introduction

With the design and development of smart cities, opportunities and challenges arise at the moment. For this purpose, a huge amount of physical sensor and social media data need to be obtained. Nevertheless, circumstance vary in different cities due to the variant infrastructures and populations, which leads to the data sparsity, Bassoli et al. [1]. For example, because of the huge population and perfect infrastructure, social media data in big cities are relatively easy to obtain. However, small towns have smaller populations and, hence, relatively inactive social media. Therefore, it is difficult to build a smart city system based on those data. Meanwhile, numerous applications have been modeled through the analysis of data in large cities. To this end, we transfer knowledge from big cities to small ones for urban waterlogging disaster analysis.

With the increasing severity of urban waterlogging disasters in some developing countries, such as China and India, urban waterlogging analysis has become a critical component in modern smart city systems, Gupta [2] and Zhang et al. [3]. Accurate analysis of urban waterlogging conditions can significantly help traffic management agencies generate proactive strategies to mitigate congestion, which can help drivers better plan their trips by avoiding routes expected to be congested. Existing research in the area primarily focuses on past and current conditions, as well as sensors and cameras. However, these data are relatively insufficient to plan for the entire city. Thus, there is considerable interest in using social media to detect urban waterlogging without using physical sensors.

With the rapid growth of social media, more and more people are using Twitter, Facebook, and so forth, to communicate their moods, activities, and plans, as well as to exchange news and ideas, Cranshaw et al. [4]. This has created a massive repository containing information inaccessible through conventional media. This repository includes users' messages relating to urban waterlogging conditions in their areas at different times, such as "Deep water on new seven street. Cars unmoved" and "Big road blocks at intersection; tire in deep water". In large cities, it is viable to have at one's disposal large amounts of data related to urban waterlogging, Yin et al. [5], Quan et al. [6], and Yadav et al. [7]. However, small cities may not produce adequate social media data for this purpose.

Moreover, most waterlogging events are caused by poor road networks, low terrains, and high precipitation in a short time, and normally these kinds of data are easily obtained. Moreover, data in different cities have different distributions. For example, different people may post different tweets for the same event because of reginal differences. The same physical condition may also not necessarily lead to the same severity of waterlogging. In such cases, different cities are equivalent to different domains. It would be helpful if we could transfer urban waterlogging knowledge from its local domain to a new one.

Motivated by the uniqueness of the information available on social media and through satellites and the close relationship between this information and the severity of urban waterlogging, we set ourselves the task of determining whether we can retrieve the relevant Twitter and satellite data and transfer the knowledge conveyed by these to small cities to analyze urban waterlogging, Wu et al. [8]. We analyze twitter data to acquire the locations of urban areas affected by waterlogging and determine the severity. We utilize the open APIs to access the stream of observation records and then establish a correlation between the relevant social media content and satellite features. Moreover, we map the location with entities from external knowledge bases to enrich features. Following this, we analyze the waterlogging data and transfer them to small cities, for which we do not have adequate data of this sort, through a multiview discriminant transfer learning method. We found that most small cities can monitor urban waterlogging disasters through our method.

This paper's major contributions can be described in three aspects:

(1) We propose a multiview discriminant transfer learning method between cities for urban waterlogging disaster analysis.

(2) We analyze the features that have influence on urban waterlogging disaster analysis.

(3) We evaluate the method by various data sources including global satellite-based precipitation data, weather forecast reports data, and Weibo/WeChat data in China and India.

The rest of the paper is organized as follows. In Section 2, we briefly review existing work on social media disaster mitigation and data sparsity in urban computing. We offer preliminary definitions and present the problem statement in Section 3. In Section 4, we propose social and physical view analyses as well as the proposed multiview discriminant transfer learning method for urban waterlogging. We show the setup and results of our experiments in Section 5 and conclude the paper in Section 6.

2. Related Work

2.1. Social Media for Disaster Mitigation. As mentioned in Section 1, researchers nowadays are trying to exploit the wealth of information available on social media for various purposes. For example, there is considerable interest in using social media to detect emerging news or events: in Petrović et al. [9], the authors address the problem of detecting new events from a stream of Twitter posts using an algorithm based on locality-sensitive hashing. In Sankaranarayanan et al. [10], the authors propose a new processing system called "TwitterStand" to capture tweets that correspond to breaking news. The authors in Sakaki et al. [11] investigate the real-time reception of events, such as earthquakes, on Twitter, and propose a probabilistic spatiotemporal model for the target event that can locate the center and the trajectory of the event.

Furthermore, some researchers are investigating the extraction from tweets of information that might be useful in other domains. In Bollen et al. [12], the authors attempted to determine whether public mood correlates with, or is even predictive of, economic indicators. To this end, they first derived collective mood states from large-scale Twitter feeds and then performed a correlation analysis with the Dow Jones Industrial Average (DJIA) over a certain period of time. They showed that the accuracy of DJIA predictions can be significantly improved by including specific public mood dimensions, such as "calm." In Eisenstein et al. [13], based on geotagged social media, the authors proposed a multilevel generative model that reasons jointly about latent topics and geographical regions.

With the revival of interest in deep learning, incorporating the continual representation of a word as a feature has proved to be effective in a variety of natural language processing (NLP) tasks, such as parsing, language modeling, and named entity recognition (NER). In sentiment analysis, Bespalov et al. [14] initiated word-embedding using Latent Semantic Analysis and represented each of several documents as the linear weight of n-gram vectors for sentiment classification. Our proposed work belongs to this direction of research, and we attempt to build a correlation between Twitter data and a new domain, namely, urban waterlogging analysis.

2.2. Data Sparsity in Urban Computing. The problem of data-missing was caused by many reasons. For example, different venues have different user visits. More seriously, some venues may not have people visiting them at all. Data sparsity has been studied for many years in research. In urban computing, there have been many techniques that can be applied to tackle this problem. Matrix factorization decomposes a matrix into a production of two or three matrices. When the matrix is very sparse, we usually can approximate it with three low-rank matrices. For more dimensions, tensor decomposition can be used to approximate the tensor with the multiplication of three low-rank matrices and a core tensor. However, these methods can only handle data sparsity in a single city.

There has been a major assumption that the training and testing data must be in the same feature space in machine learning tasks. Nevertheless, this assumption may not hold in many real-world applications, Pan and Yang [15]. For instance, in a classification task, we have insufficient data in one domain of interest but we only have sufficient training data in another domain of interest, which follow a different distribution. Luckily, transfer learning algorithms help solve this problem, which also can deal with data sparsity problems in urban computing.

Transfer learning models data from related but not identically distributed sources. Multiview learning has been studied extensively in single-domain settings, such as cotraining, Dai et al. [16]. However, little has been done with regard to multiview transfer. Chen et al. [17] proposed Cotraining for Domain Adaptation (CODA), a pseudo multiview algorithm with only one view for original data that may not be effective for the real multiview case. Zhang et al. [18] proposed an instance-level multiview transfer algorithm (MVTL-LM) that integrates classification loss and views consistency terms in a large-margin framework. Yang and Gao [19] proposed a multiview discriminant transfer (MDT) learning approach for domain adaptation. Unlike MVTL-LM, our method is at the feature level, which mines the correlations between views together with the domain distance measure to improve transfer. Unlike MDT, our method additionally labels data by social media and is optimized in a mapping algorithm in the urban waterlogging analysis case.

3. Approach

3.1. Preliminary

Definition 1 (city block). City block is a region divided by a city (e.g., 1 km × 1 km in our experiments), assuming that urban waterlogging severity in different blocks g is uniform.

Definition 2 (social view). A feature vector svi of a block which is obtained by various social media assistance data analyses of smart cities:

$$\text{svi} = f(\text{tweets}), \tag{1}$$

where $f(\cdot)$ is function that converts the block's raw social media data into a feature vector of social view and tweets are the twitter and Weibo (a twitter like website in China) texts with geotags posted in this block.

Definition 3 (physical view). A feature vector pvi of a block obtained from physical sensors:

$$\text{pvi} = g(\text{precipitation}, \text{terrain}, \text{POIs}, \text{road}), \tag{2}$$

where $g(\cdot)$ is function that converts the block's raw physical sensor data into a feature vector of physical view and precipitation, terrain, POIs, and road are the raw data of precipitation, terrain, POIs, ad road obtained in this block.

3.2. Framework.

As shown in Figure 1, our framework consists of two major parts: feature extraction of the original city and transfer learning, which involves the analysis of urban waterlogging in small cities. We also map the locations of blocks with entities from Yago2 (http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/), geoname (http://www.geonames.org/), and WikiData (http://www.wikidata.org) to enrich our waterlogging related knowledge. For example, we may obtain a POI category "Tiandu" for "Residential district." In this way we may be able to obtain additional knowledge about "Tiandu" and "Residential district"; for example, "what was the place Tiandu before (a lake or lowland may result in severe waterlogging)," or "where is the nearest river to the residential district." We construct the social view and physical view separately and through multiview discriminant transfer learning we transfer urban waterlogging knowledge to small cities.

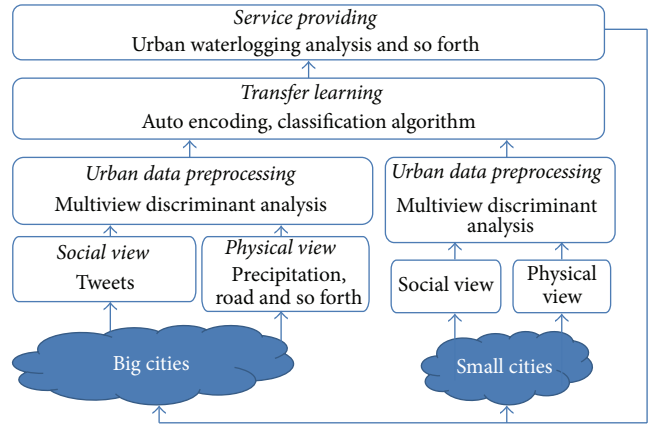


Figure 1: Multiview discriminant transfer learning framework.

Problem Statement. Each city contains blocks $C_{si} = \{D_s\}$. We use urban sensors and news reported events of each block's waterlogging severity as label. A three-level rating system is adopted: normal (there is no waterlogging at all), middle (the water is very shallow and has no effect on driving), and severe (sever road waterlogging and dangerous for driving). For example, if the sensors or news reporting the location "Changle Road, Hangzhou" have severe urban waterlogging, we calculate the block's id of this location and get the label data "2_15 2", where the first "2" is the id of "Hangzhou," "15" is the id of block that contains "Changle Road,"and the last "2" is the label. For the source cities, we have $D_s = \{(\text{svi}^{(i)}, \text{pvi}^{(j)}, y^{(k)})\}$, where $\text{svi}^{(i)}$ and $\text{pvi}^{(j)}$ are social view and physical view of block i, $y_i \in \{0, 1, 2\}$.

We utilize FDA2, Diethe et al. [20], to learn a "middle" feature representation f_i of a block i from $\text{svi}^{(i)}$ and $\text{pvi}^{(i)}$. Then we follow the research of autoencoders, Zhuang et al. [21], to build a feature mapping and use the source domain data jointly for classifier training.

4. Transfer Learning Framework

4.1. Model Social View.

The social media obviously will have huge amount of waterlogging related data for different blocks. So we obtain tweets from twitter and Weibo to obtain features to analysis of urban waterlogging in Indian and China.

We use the trained word-embedding from Glove, Pennington et al. [22]. We manually construct a dictionary D of urban waterlogging with its severity description phrases like {"seeseaontheraod", "deepwater, carunmoved", ...} and label the phrases. Then we can calculate the average vectors of phases describing normal, middle, and severe waterlogging: $\text{Vec}_{\text{normal}}, \text{Vec}_{\text{midlle}}, \text{Vec}_{\text{severe}}$. Furthermore, we calculate the top 50 words $W[50] = \{\text{word}_1, \text{word}_2, \ldots, \text{word}_{50}\}$ that

appear in D most. For a certain block, now we obtain each word-embedding of tweets posted in this region. We assume that each waterlogging related tweet is true; then we construct the feature vector of social view related to the severity of urban waterlogging. The distance between the words or phases describing the severity can be a good measurement of the real severity of urban waterlogging. For example, some phrases often appear in the place of serious waterlogging such as "see the sea in the downtown," which means the waterlogging is serious. We represent word-embedding of phases with the average of its word vectors. The word-embedding of phases near these phases describes almost the same severity. Moreover, the frequency of words is also a good feature of event severity. For instance, a block with more than 100 severe waterlogging related tweets may truly have severe waterlogging. Specifically, for a tweet "deep water, car unmoved," we firstly calculated the vector of tweet through $\text{vec}_{\text{tweet}} = (\text{vec}(\text{"deep"}) + \text{vec}(\text{"water"}) + \text{vec}(\text{"car"}) + \text{vec}(\text{"unmoved"}))/4$; then we calculate the distance of $\text{vec}_{\text{tweet}}$ and $\text{Vec}_{\text{normal}}, \text{Vec}_{\text{midlle}}, \text{Vec}_{\text{severe}}$, respectively. At last, we observe weather the words in $W[50]$ appear in tweet and record the times.

Finally we build social view of block as

$$\text{svi} = \left[\frac{\sum_1^N \left(\text{vec}_i - \text{Vec}_{\text{normal}}\right)}{N}, \frac{\sum_1^N \left(\text{vec}_i - \text{Vec}_{\text{midlle}}\right)}{N}, \frac{\sum_1^N \left(\text{vec}_i - \text{Vec}_{\text{severe}}\right)}{N}, \text{Appearance}\,[50] \right], \tag{3}$$

where vec_i means the word-embedding of a tweet, N is the number of tweets in this block, $\text{Vec}_{\text{normal}}, \text{Vec}_{\text{midlle}}, \text{Vec}_{\text{severe}}$ are average vectors of phases describing normal, middle, and severe waterlogging, and Appearance[50] is a 50-dimensional vector and records the occurrence number of words in W (Appearance[1] = 12 means the word $W[1]$ appears 12 times in the tweets from this block).

4.2. Model Physical View. The concentration of urban waterlogging is influenced by meteorology and terrain. Accordingly, we identify precipitation. We analyze the correlation matrix between urban waterlogging severity and such features using data collected from several cities in China and India. More specifically, for each waterlogging location l_i, we measure (1) precipitation, (2) terrain, (3) road network, and (4) POIs by mining the physical features in g_i, $\{p : p \in P \& p \in g_i\}$ in which P is the set of physical view in big cities.

4.2.1. Precipitation. Apparently, the precipitation in specific area and specific period imply the locations and severity of waterlogging. We use the total precipitation data in the last one, two, three, six, twelve, and twenty-four hours and time in the block as features. Formally we have

$$f_i^{PR} = \left[\left(P_1, P_2, P_3, P_6, P_{12}, P_{24}, t\right) \right], \tag{4}$$

where P_i is the precipitation in last i hours and t is the current time.

4.2.2. Terrain. Apparently, a high terrain disperses the concentration of severity, and high precipitation usually causes a high concentration. For example, for a block g_i, it has eight neighbours q_i. We have to calculate the relative terrain value of block g_i in consideration of q_i. For example, a place may have a low terrain value (normally have high possibility of occurrences of waterlogging), but its neighbors are much lower in terrain values, so the possibility of occurrences of urban waterlogging is lower. Formally we have

$$f_i^T = \left[\left(\sum_{j=1}^{8} \left(T\left(q_j\right) - T\left(g_i\right) \right), \lambda \right) \right], \tag{5}$$

where q_j is in the block next to g_i, $T(q_j)$ means the elevation of block q_j, and λ is the elevation measurement error.

4.2.3. Road Network. The structure of a road network has a strong correlation with its terrain pattern, thus providing a satisfactory complement to severity modeling. We identify the following three features for each block based on a road network database: (1) length of elevated road, (2) number of culverts, and (3) the number of intersections in the region. Formally we have

$$f_i^{RN} = \left[(L, C, I) \right], \tag{6}$$

where I is the number of intersections, L is the length of elevated road, and C is the number of culverts.

4.2.4. POIs. The POIs indicate the patterns of this region, hence contributing to urban waterlogging analysis. A POI may have directly causal relation to it. For example, if a region has large building land spaces, its severity tends to be bad. A park, however, usually leads to less waterlogging. In short, these features are significantly discriminative in urban waterlogging severity analysis. Hence, we apply an entropy to measure the functionality heterogeneity of a block. Let $\sharp(i, c)$ denote the number of POIs of category $c \in C$ located in g_i and $\sharp(i)$ be the total number of POIs of all categories located in g_i. The entropy is defined as

$$f_i^{\text{POI}} = -\sum_{c \in C} \frac{\sharp(i, c)}{\sharp(i)} \times \log \frac{\sharp(i, c)}{\sharp(i)}. \tag{7}$$

At last, we have

$$\text{pvi} = \left[f_i^{PR}, f_i^T, f_i^{RN}, f_i^{\text{POI}} \right]. \tag{8}$$

4.3. Multiview Discriminant Analysis. In reality we now can concatenate social view and physical view into one single view to adapt to the learning setting. However, this concatenation causes overfitting in the case of a small size training sample and is not physically meaningful because each view has a specific statistical property. According to studies of Zheng et al. [23], treating features extracted from different data sources equally do not achieve the best performance. In contrast to single view learning, multiview learning as a new paradigm introduces one function to model a particular

view and jointly optimizes all the functions to exploit the redundant views of the same input data and improve the learning performance.

Diethe et al. extended Fisher's Discriminant Analysis (FDA) to FDA2 by incorporating labeled two-view data into the Canonical Correlation Analysis (CCA) framework as follows, Diethe et al. [20] and Melzer et al. [24]:

$$\max_{(w_s,w_p)} \frac{w_s^T M_w w_p}{\sqrt{w_s^T w_s} \cdot \sqrt{w_p^T M_p w_p}}, \tag{9}$$

where

$$\begin{aligned} M_w &= X_s^T y y^T Z_s, \\ M_s &= \frac{1}{n} \sum_{i=1}^{n} \left(\phi\left(s_i\right) - \mu_s\right) \left(\phi\left(s_i\right) - \mu_s\right)^T, \\ M_p &= \frac{1}{n} \sum_{i=1}^{n} \left(\phi\left(p_i\right) - \mu_p\right) \left(\phi\left(p_i\right) - \mu_p\right)^T, \end{aligned} \tag{10}$$

where w_s and w_p are the means of the source data from the two views. The numerator in (9) reflects the interclass distance, which needs to be maximized, while the denominator reflects the intraclass distance, which should be minimized. The above optimization problem is equivalent to selecting vectors that maximize the Rayleigh quotient:

$$r = \frac{\zeta^T Q_w \zeta}{\zeta^T P \zeta}, \tag{11}$$

where $Q_w = \begin{pmatrix} 0 & M_w \\ M_w^T & 0 \end{pmatrix}$, $P = \begin{pmatrix} M_s & 0 \\ 0 & M_p \end{pmatrix}$ and $\zeta = \begin{pmatrix} w_s \\ w_p \end{pmatrix}$. Note that Q_w encodes the interclass distance, whereas P encodes the compound information about the view-based intraclass distances. Further, ζ is an eigenvector. Such an optimization is different from FDA2 and facilitates its extension to cross-domain scenarios, which will be presented in the following subsection. For an unlabeled instance, the classification decision function is given by

$$f\left(s_i, p_i\right) = \left[w_s^T \phi\left(s_i\right) + w_p^T \phi\left(p_i\right) - b\right], \tag{12}$$

where b is the threshold.

After multiview discriminant analysis, we obtain a single view feature vector for each city, which is able to be used by machine learning algorithm. However, the sparsity of label data is still a problem. The model based on these single city data is quite unreliable. So we try to use transfer learning.

4.4. Autoencoders. In our problem, the feature vector of different cities has different distributions. The feature-representation-transfer approach to the inductive transfer learning problem aims at finding good feature representations to minimize domain divergence and classification or regression model error. We use autoencoder to construct a feature representation. An autoencoder is a mapping from an instance x to a hidden representation z through $z = h(Wx + b)$, After that, the hidden representation z is reconstructed to $\widehat{x} = g(W'z + b')$.

The object function of autoencoder is formalized as

$$\min_{W,b,W'b'} = \sum_{i=1}^{n} \left\| x_i - \widehat{x}_i \right\|^2. \tag{13}$$

4.5. Transfer Learning. Finally we proposed the optimization problem as follows:

$$\begin{aligned} \min_{\Theta,\Theta',\{\theta_j\}} = {} & \epsilon\left(x_S, \widehat{x}_S, x_T, \widehat{x}_T\right) + \gamma\Omega\left(\Theta, \Theta'\right) \\ & + \alpha\imath\left(z_S, y_S; \left\{\theta_j\right\}\right), \end{aligned} \tag{14}$$

where the first term represents the reconstruction error:

$$\begin{aligned} \epsilon\left(x_S, \widehat{x}_S, x_T, \widehat{x}_T\right) = {} & \sum_{i=1}^{r} \sum_{i=1}^{n_j} \left\| x_{S_i} - \widehat{x}_{S_i} \right\|^2 \\ & + \sum_{i=1}^{n} \left\| x_{T_i} - \widehat{x}_{T_i} \right\|^2, \end{aligned} \tag{15}$$

where x_S, x_T are the source and target feature representations and $\widehat{x}_S$, $\widehat{x}_T$ are the representations through encoding and decoding by autoencoder.

The second term represents regularization: $\Theta = \{W, b\}$ and $\Theta' = \{W', b'\}$.

The third term represents the total loss of the softmax regression classifier.

We adopt the gradient descent methods for the solution.

Algorithm 4. Multiview transfer learning with autoencoders:

Input

The source dataset $D_s = \{(\text{svi}^{(i)}, \text{pvi}^{(j)}, y^{(k)})\}$

The target dataset $D_t = \{(\text{svi}^{(m)}, \text{pvi}^{(n)}, y^{(p)})\}$

Trade-off parameters α, γ, hidden features number k.

Output. Target domain classifier.

(1) Initialize W, b, W', and b'.

(2) Run multiview discriminant analysis to combine social view svi and physical view psi into a single view.

(3) Fix $\{\theta_j\}$; update W, b, W', and b' alternatively.

(4) Fix W, b, W', and b'; update $\{\theta_j\}$.

(5) If converge, output classifier; otherwise, go to Step (3).

4.6. Target Classifier Construction. Given b, W', b', W, and $\{\theta_j\}$, the classifier f_T can be obtained. Formally, we have

$$f_T\left(x_T\right) = \frac{1}{r} \sum_{j=1}^{r} \left(\theta_j^T \left(g\left(W x_T + b\right)\right)\right), \tag{16}$$

where $g(x)$ is the classifier function of softmax regression.

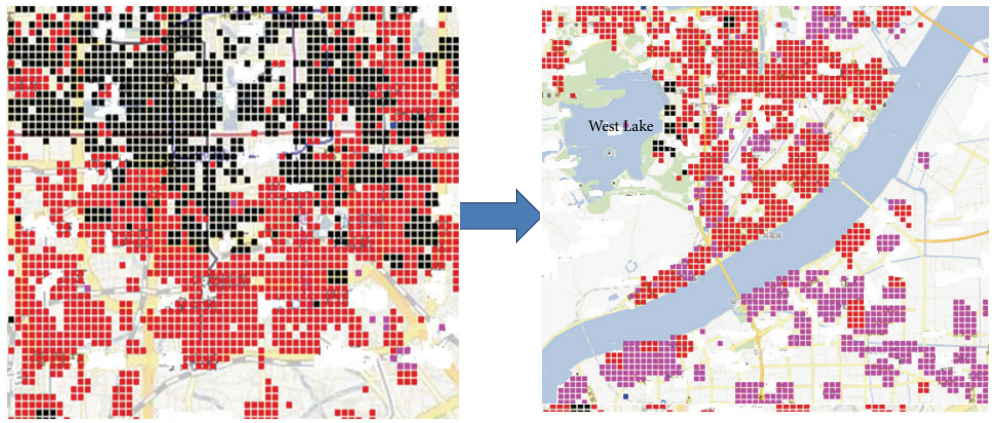


Figure 2: Social media in Beijing and Hangzhou.

Table 1: Details of the datasets.

Data sources	China	India
City number	10	4
Social media	35,324,237 (Weibo)	1,321,421 (Twitter)
Precipitation	2013.5–2013.12 (7 M)	2013.5–2013.12 (3 M)
Road network	534 (Culvert)	231 (Culvert)
Terrain	437	253
POIs	310,230	75,217

5. Experiments

5.1. Datasets. Urban waterlogging is one of the most serious hazards in several big cities across the world, especially in China and India. In 2013, hundreds of cities reported being waterlogged for dozens of days at various times. The source code and sample data of experiments can be obtained from https://github.com/zxlzr/UrbanWaterloggingInference. In our experiment, we used the following five real datasets showed in Table 1:

(1) Social media: we collected data from both Twitter and Sina Weibo, which is a twitter-like website that was in use across at least 10 cities in China and India in 2013 and 2014.

(2) Meteorological data: we collected precipitation meteorological data from the National Oceanic and Atmospheric Administration's (NOAA) web service every hour.

(3) POIs: we collected POI data from Baidu Maps for each city.

(4) Road networks: the road network data was gathered from Openmaps.

(5) Terrain: the terrain data was from Openmaps as well.

The feature data distributions we used in different cities are quite different. As Figure 2 shows, the dark regions mean the waterlogging related social media data are massive. The social media data in Beijing is far more than Hangzhou. Moreover, the label sparsity is quite different. For example, 121 blocks in Beijing (total 200) once have severe urban waterlogging record; however for some relatively small cities such as Hangzhou (total 180) the record is only 65. So we use transfer learning to combine more data. For Hangzhou as a target city and Beijing as source city, we use 30 blocks in Hangzhou and 121 blocks in Beijing of label data as training data, the other 35 as testing data.

5.2. Evaluation. In order to get the highest accuracy for all the models, we cross-validate using the development set to find the best hyperparameters. We obtain free-text descriptions of places by adopting geoparsing (https://github.com/ropenscilabs/geoparser) to convert text into unambiguous geographic identifiers (lat-lon coordinates) and map the entities with external knowledge bases. We set the trade-off parameters $\alpha = 0.01$, $\gamma = 0.03$, and the number of hidden features $k = 100$.

We use different baseline algorithms to verify the effectiveness of our method as follows:

(1) *GBRT.* Gradient boosting is a machine learning technique for regression and classification problems. We only use single city data to build a model.

(2) *ANN.* We choose Artificial Neural Network (ANN) with backpropagation technique as another baseline. The constructed ANN contains one hidden layer. The ANN method simply treats all labeled data from all stations and all cities (not using multiview) as the training data to build a model.

(3) *TrAdaBoost.* Dai et al. [25] proposed a boosting algorithm, TrAdaBoost, which is an extension of the AdaBoost algorithm, to address the inductive transfer learning problems. It attempts to iteratively reweight the source domain data to reduce the effect of the bad source data while encouraging the good source data to contribute more for the target domain and it is an instance-transfer approach.

Metrics. Because we use a three-level rating system ($2 > 1 > 0$), given ReferanceSet obtained from sensors and news

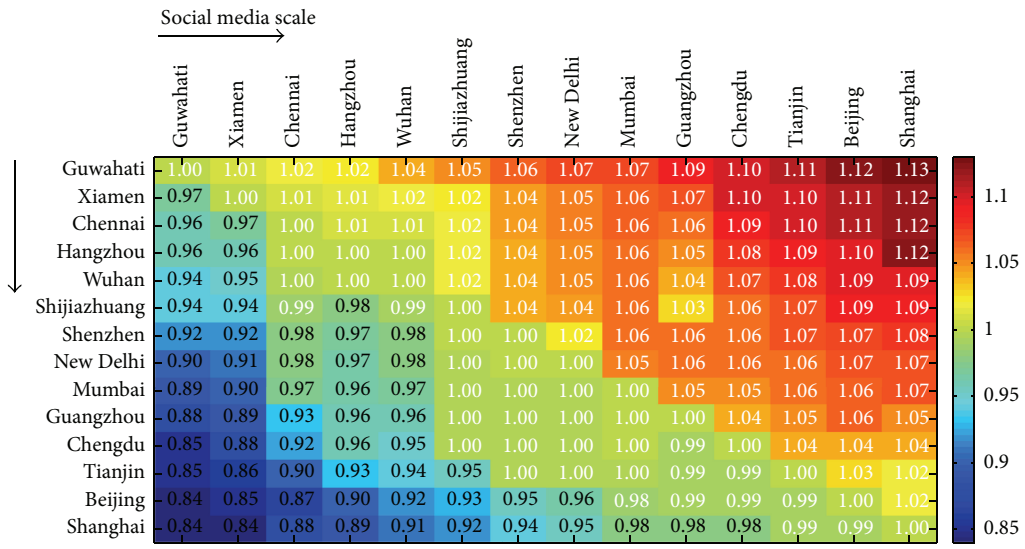


Figure 3: Transfer learning from a single city.

report and given PredictionSet = $\{y_i \geq 1\}$, formally we have

$$\text{Precision} = \frac{\left|\bigcap(\text{PredictionSet}, \text{ReferanceSet})\right|}{|\text{PredictionSet}|},$$
$$\text{Recall} = \frac{\left|\bigcap(\text{PredictionSet}, \text{ReferanceSet})\right|}{|\text{ReferanceSet}|}. \tag{17}$$

We evaluate the final result with $F1 = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$ score.

5.3. Results

5.3.1. Single City Transfer. We chose a source city and transferred the relevant waterlogging knowledge pertaining to it to a target city. The enhancement obtained by the transfer of learning over the original method is showed in (18). Note that F_{st} represents the $F1$ score of transfer learning methods from city s to city t, whereas F_t is the $F1$ score obtained by city t itself by GBRT.

In Figure 3, we show the difference in enhancement between the transfer learning method and the method obtained directly from the city. Different boxes represent the speedup of the transfer learning method and the direct method, from the city represented in the abscissa to that in the ordinate. With the increase in the coordinate axis, the social media size of the city increases, which means the social media size in Beijing is bigger than Tianjin, for example. We use the social media size to evaluate its relative size of the city. In fact, city with more social media activities is usually relatively bigger than that with less social media activities. We see that our transfer learning method outperformed the method obtained directly from the city when transferring knowledge from a large city to a small one. Actually, with the size of social media increasing, the size of training data increases.

$$\text{Enhancement}_t = \frac{F_{(st)}}{F_t}. \tag{18}$$

Table 2: The results of baseline and transfer learning for five target cities.

Cities	Social view	Physical view	Total
		GBRT	
Beijing	0.745	0.613	0.815
Shanghai	0.732	0.589	0.712
Hangzhou	0.555	0.511	0.711
Mumbai	0.781	0.631	0.811
New Delhi	0.781	0.711	0.722
		ANN (not multiview)	
Beijing	0.742	0.675	0.802
Shanghai	0.715	0.688	0.803
Hangzhou	0.713	0.566	0.814
Mumbai	0.832	0.662	0.802
New Delhi	0.754	0.744	0.813
		TrAdaBoost (instance-transfer)	
Beijing	0.762	0.676	0.812
Shanghai	0.755	0.617	0.883
Hangzhou	0.733	0.555	0.824
Mumbai	0.832	0.722	0.852
New Delhi	0.724	0.784	0.862
		Multiview transfer learning with autocoder (feature-transfer)	
Beijing	0.753	0.638	**0.852**
Shanghai	0.755	0.687	0.863
Hangzhou	0.763	0.595	**0.894**
Mumbai	0.832	0.732	**0.892**
New Delhi	0.754	0.794	**0.912**

5.3.2. Multiple City Transfer. Finally, we attempted to transfer knowledge from multiple cities to a single city. For example, if Beijing is a target city, we use total cities' data as source city (14 cities). In Table 2, we present the results for all features. Actually, we have observed an improvement compared with

the baseline. Regardless of any method, the $F1$ score using total data is better than only using social view or physical view. The GBRT method uses only single city data; its effeteness is the worst. ANN only concentrate all data and do not use multiview method. It is not as good as multiview method. TrAdaBoost and our method use multiview method, while TrAdaBoost is an instance level transfer learning. However, in our problem, the features in different cities have different distribution. So feature-transfer method is better.

6. Conclusions

In this paper, we analyzed urban waterlogging using four datasets. We transferred waterlogging knowledge between cities and evaluate our method over 10 cities and a period of more than 18 months. The evaluation showed that transferring urban waterlogging to small cities is applicable.

The transfer learning algorithm proposed here may also have the same effect for some kind of data sparsity, such as AQI in small cities. Thus, we hypothesize that the our algorithm will succeed in transferring other urban knowledge, such as air pollution and traffic from larger to smaller cities and towns with scarce data. This can be understood by analyzing the difference between cities and the rich knowledge transfer obtained from big cities.

In future, we would like to apply our approach to more cities. In addition, we would like to use transfer learning method to solve other data sparsity problems in machine learning.

Competing Interests

The authors declare no conflict of interests.

References

[1] A. Bassoli, J. Brewer, K. Martin, P. Dourish, and S. Mainwaring, "Underground aesthetics: rethinking urban computing," *IEEE Pervasive Computing*, vol. 6, no. 3, pp. 39–45, 2007.

[2] K. Gupta, "Urban flood resilience planning and management and lessons for the future: a case study of Mumbai, India," *Urban Water Journal*, vol. 4, no. 3, pp. 183–194, 2007.

[3] X. Zhang, M. Hu, G. Chen, and Y. Xu, "Urban rainwater utilization and its role in mitigating urban waterlogging problems—a case study in Nanjing, China," *Water Resources Management*, vol. 26, no. 13, pp. 3757–3766, 2012.

[4] J. Cranshaw, R. Schwartz, J. I. Hong, and N. Sadeh, "The Livehoods project: utilizing social media to understand the dynamics of a city," in *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media (ICWSM '12)*, pp. 58–65, June 2012.

[5] Z. Yin, J. Yin, S. Xu, and J. Wen, "Community-based scenario modelling and disaster risk assessment of urban rainstorm waterlogging," *Journal of Geographical Sciences*, vol. 21, no. 2, pp. 274–284, 2011.

[6] R.-S. Quan, M. Liu, L.-J. Zhang et al., "Exposure assessment of rainstorm waterlogging on buildings in central urban area of shanghai based on scenario simulation," *Chinese Geographical Science*, vol. 2, pp. 148–152, 2011.

[7] K. Yadav, D. Chakraborty, S. Soubam et al., "Human sensors: case-study of open-ended community sensing in developing regions," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom '13)*, pp. 389–392, March 2013.

[8] S. G. Wu, M. T. Zhou, and D. D. Ma, "Application of imitated ecosystems in urban waterlogging prevention," in *Applied Mechanics and Materials*, vol. 587–589, pp. 501–504, Trans Tech Publications, 2014.

[9] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Proceedings of the Human Language Technologies Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT '10)*, pp. 181–189, Association for Computational Linguistics, Los Angeles, Calif, USA, June 2010.

[10] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling, "TwitterStand: news in tweets," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 42–51, ACM, Seattle, Wash, USA, November 2009.

[11] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," in *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*, pp. 851–860, ACM, Raleigh, NC, USA, April 2010.

[12] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

[13] J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing, "A latent variable model for geographic lexical variation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pp. 1277–1287, Association for Computational Linguistics, 2010.

[14] D. Bespalov, B. Bai, Y. Qi, and A. Shokoufandeh, "Sentiment classification based on supervised latent n-gram analysis," in *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM '11)*, pp. 375–382, ACM, Glasgow, UK, October 2011.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[16] W. Dai, Y. Chen, G.-R. Xue, Q. Yang, and Y. Yu, "Translated learning: transfer learning across different feature spaces," in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS '08)*, pp. 353–360, Vancouver, Canada, December 2008.

[17] M. Chen, K. Q. Weinberger, and J. C. Blitzer, "Co-training for domain adaptation," in *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS '11)*, pp. 2456–2464, Granada, Spain, December 2011.

[18] D. Zhang, J. He, Y. Liu, L. Si, and R. D. Lawrence, "Multi-view transfer learning with a large margin approach," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 1208–1216, ACM, San Diego, Calif, USA, August 2011.

[19] P. Yang and W. Gao, "Multi-view discriminant transfer learning," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, AAAI Press, pp. 1848–1854, Beijing, China, August 2013.

[20] T. Diethe, D. R. Hardoon, and J. Shawe-Taylor, "Multiview fisher discriminant analysis," in *Proceedings of the NIPS Workshop on Learning from Multiple Sources*, 2008.

[21] F. Zhuang, X. Cheng, S. J. Pan, W. Yu, Q. He, and Z. Shi, "Transfer learning with multiple sources via consensus regularized autoencoders," in *Machine Learning and Knowledge Discovery in Databases*, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds., vol. 8726 of *Lecture Notes in Computer Science*, pp. 417–431, 2014.

[22] J. Pennington, R. Socher, and C. Manning, "Glove: global vectors for word representation," in *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP '14)*, pp. 1532–1543, Doha, Qatar, October 2014.

[23] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: when urban air quality inference meets big data," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*, pp. 1436–1444, ACM, Chicago, Ill, USA, 2013.

[24] T. Melzer, M. Reiter, and H. Bischof, "Appearance models based on kernel canonical correlation analysis," *Pattern Recognition*, vol. 36, no. 9, pp. 1961–1971, 2003.

[25] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 193–200, Corvallis, Ore, UA, June 2007.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 3506261, 11 pages
http://dx.doi.org/10.1155/2016/3506261

Research Article

Fracture Mechanics Method for Word Embedding Generation of Neural Probabilistic Linguistic Model

Size Bi, Xiao Liang, and Ting-lei Huang

Institute of Electronics, Chinese Academy of Sciences, Beijing, China

Correspondence should be addressed to Ting-lei Huang; tlhuang@mail.ie.ac.cn

Received 3 January 2016; Revised 16 July 2016; Accepted 26 July 2016

Academic Editor: Trong H. Duong

Copyright © 2016 Size Bi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Word embedding, a lexical vector representation generated via the neural linguistic model (NLM), is empirically demonstrated to be appropriate for improvement of the performance of traditional language model. However, the supreme dimensionality that is inherent in NLM contributes to the problems of hyperparameters and long-time training in modeling. Here, we propose a force-directed method to improve such problems for simplifying the generation of word embedding. In this framework, each word is assumed as a point in the real world; thus it can approximately simulate the physical movement following certain mechanics. To simulate the variation of meaning in phrases, we use the fracture mechanics to do the formation and breakdown of meaning combined by a 2-gram word group. With the experiments on the natural linguistic tasks of part-of-speech tagging, named entity recognition and semantic role labeling, the result demonstrated that the 2-dimensional word embedding can rival the word embeddings generated by classic NLMs, in terms of accuracy, recall, and text visualization.

1. Introduction

Word embedding is a word numerical representation that uses a continuous vector space to represent a group of words [1]. In the word vector space, each word corresponds to a unique point. Intuitively, those points that have similar meanings should be put close, while those who are distant in meaning should be put far away. Based on the space, the degree of relation between words can be estimated via computing the distance between vector points, such as the Euclidian or Mahalanobis distance. Such semantic numeralization enables textual and symbol data to be processed in the traditional neural models. For this reason, more nature language processing (NLP) systems are improved with these neural network technologies [2] and enhance the performance of common natural linguistic tasks [3], such as POS (part-of-speech tagging), chunking, NER (named entity recognition) [4], and SRL (semantic role labeling). However, the training for such word embedding is time consuming and required sophisticated tuning for parameters of the model [5], because these models have a huge number of parameters needed for training, which are high dimensional and sparse. Moreover, text visualization for such high-dimensional data suffered the distortion of information from the preprocessing in dimensional reduction, confusing the understanding of data relations [6].

The word vectorization has always been an active topic for text mining and NLP. Their generating approaches can be roughly divided into two categories: manual coding and autocoding. Manual coding is to code words according to the knowledge of domain experts. It is a heavy work for considering the value of each word. For example, WordNet is a huge project whose aim is to build a word graph database by language experts, where word meanings are associated and presented via the tree-based structural graph [7, 8]. Such representation can only associate a few ranges of words for each word. It is insufficient to build a global relation for all words. On the other hand, autocoding is to code word by neural network models [9]. Every word is initialized with a random vector and varied following the parameter tuning according to a range of contexts around a word. Generally, such methods are performed by the training of NLM, where word embedding is a part of result when the convergence of the objective function is finished [1]. However, the NLM-based approaches such as feedforward neural networks [1],

recurrent neural network (RNN) [8], and restricted Boltzmann machine (RBM) [10] have also suffered from the high learning complexity [11], sophisticated preferences [12], and the curse of dimensionality [13].

In this article, we present a novel method for word embedding learning that can reduce the high dimensionality, which is inherent in the traditional NLM. We assume that the generation of word embedding can be viewed as a particle movement on a plane. Particles that are close represent the corresponding words have the similar meaning, whereas the particles that are distant represent the corresponding words are far away in meaning. For simulating text semantics as correctly as possible, a fracture mechanics model is presented for controlling the generating process of word embedding. We aim to provide an effective, intuitive approach for learning a 2-dimensional word vector, which is applicable to the general natural language tasks. In particular, we omit the homonymy and polysemy for keeping the consistency of word representation. With this context, each word corresponds to a single vector. The generating model that is based on neural networks is substituted by a particle system, which can simulate the correlating process of semanticity between words.

Our specific contributions are as follows:

(i) We propose a force-directed model that is based on fracture mechanics to generate word embedding. A linear elastic fracture model is introduced to control the varying progress of word semantic.

(ii) We use a language model that is based on the feedforward NLM to experiment with the word embedding on the task of POS and NER and SRL, where the word embedding is the input of the language model.

(iii) The coordinates of the 2-dimensional word embedding can be used for word visualization that facilitates observing the degree of relation among words intuitively.

The next section describes the related work regarding word numerical representation. Section 3 introduces our methodology. Sections 4 and 5 give the result and discussion. Section 6 gives a conclusion and some possible works in future.

2. Background

2.1. Text Representation by Word Embedding. Choosing an appropriate data representation can facilitate the performing of a machine learning algorithm. The related methods have developed at the level of automatic features selection according to the requirement of applications [14]. As a branch of machine learning, representation learning [15] has gradually become active in some famous communities, especially to investigate knowledge extraction from some raw datasets. Text representation in natural linguistic tasks can be divided into the three levels of corpus [16, 17], paragraph [18, 19], and word [20–22]. This paper focuses on the representation learning for words. Feature words and context have been considered as the foundation for text representation learning and the constructing of a NLP system [23–25]. We follow this direction aiming at mapping word text to a vector space. Compared with the representing level of paragraph and corpus, word is the fewer granularities of semantics that is more suitable to be analyzed by a vector distance.

The idea that uses a vector space to map word meaning is proposed from [26] initially. The earlier representation learnings fail to consider the semantic measurement for the differences between words and emphasize how to quantize feature words. For example, in [27], to represent the word "dove," the first byte of the corresponding representation denotes the property of its shape that is set to "1," if the dove satisfies some conditions. The representation learning did not focus on the context feature but presents some approaches for measuring the differences regarding word semantic. The self-organizing map (SOM) [27] is employed to compute word vector distance, which uses the length to represent the neighboring degree of word meanings. Based on SOM [28], high frequency cowords are mapped to a 90-dimensional vector space. The investigation [29, 30] for SOM-based word vector applies it in the fields of NLP and data mining.

2.2. Generation of Word Embedding. Word representation has started to integrate some measuring approaches for text semantic with the developing of neural networks in probabilistic linguistic model. Reference [31] proposed to use neural network to build a language model. There exits an assumption that the numerical result of those similar meaning words should be put closely, whereas the result regarding those distant meaning words should be put far away. Meanwhile, the probability function is introduced to add the probability output for NLM, which can give a statistics result for estimating a given n-gram words combination. In the proposed models, a convoluted neural network [3] that is tuned by hand somewhat after learning obtains an accuracy of 97.20% in POS, 93.65% in chunking, 88.67% in NER, and 74.15% in SRL. A comparison between conditional restricted Boltzmann machine models and support vector machines [32] is performed for a music annotator system, which is based on the context around lyric that can use cooccurrence and sequential features to improve the accuracy of labeling. The SENNA software [3, 33] performs a word sense disambiguation with an accuracy of 72.35%. Word representation is denoted by multiple vectors [4] to express the polysemy that is tested on NER, which shows that it rivals the traditional linguistic models. In addition, word representation learning has been trained based on the n-gram models, hidden Markov models, and partial lattice Markov random field model [34].

For representation learning by NLM, [1] proposes a feedforward neural network to train the word embedding, which is regarded as the internal parameters requiring the tuning following the object function. Reference [35] presents a text sequence learning model that is called RNN, which is capable of capturing local context feature in sequence of interest. Following this route, more machine learning algorithms are introduced to improve the weaknesses of natural linguistic tasks. Reference [10] uses the restricted Boltzmann machine to improve the performance of the sequential probability function. Reference [36] creates a hierarchical learning that

Table 1: Properties of word-particle.

Name	Notes
ID	Identifier
Pos	Coordinates (word embedding)
Word	Corresponding word
Mass	Synthetic aliveness
TmpMass	All backward related particle aliveness
history_Mass	Historical aliveness
Current_Mass	Current aliveness
Base_Mass	Identifier
Chain	Backward related index
Max_flaw	Backward related degree
Flaw	Current flaw length
Radius	Repulsion radius
Pull_Radius	Tension-start radius
Velocity	Current velocity of particle

represents the semantic relation among words as a tree structure. The deep learning mechanism is tried to build a NLM [37].

2.3. Force Simulated Method. The force-directed algorithms are mainly applied in data visualization. Reference [38] compares several force-directed algorithms. Reference [39] uses these methods for analyzing a complex social network, which adds a gravity to draw the graph of social network. In some cross-domain applications, wireless sensors network uses it to build layouts [40], and [41] performs electrical circuit layouts automatically based on the force rules.

The reason why we use the force simulated approach to improve the generation of word embedding is that the relation between words semantic is relatively stable within a certain numbers of documents that depict the similar topic. This is somewhat like the convergence of the stable energy status in the force-directed algorithm. We are inspired it to use the idea of the force-related methods to improve the problems of NLM.

3. Methodology

3.1. Word-Particle Model. To map word into a particles system, we must define a mapping rule that specifies which attribute of particle corresponds to which feature of word semantics. The linking table is shown in Table 1. The table consists of two parts, where the left part designates the names of each property for particles and the right part gives the explanation for the corresponding semantic feature.

For more explanation, we explain them one by one.

(i) ID, each word has a unique identifier for relating the corresponding particle.

(ii) Pos, it is exactly the attribute that we want to train, which is also called word embedding that denotes the coordinate of a particle in a plane actually.

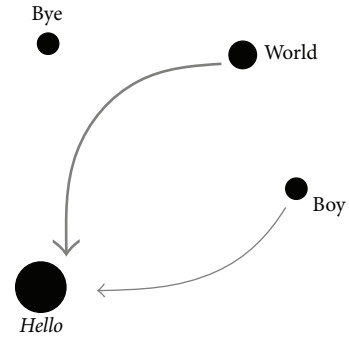


Figure 1: An example of the word-particle model.

(iii) Mass, we use the concept of particle mass to denote the occurrence frequency of a word.

(iv) TmpMass, we use the temporal mass of particle to represent the coword frequency.

(v) Current_Mass & history_Mass & base_Mass, they represent the current mass, historical mass, and basic mass of a particle, respectively. A function combined with them that is used to control the occurrence frequency of coword within a sampling period is described as

$$\text{Mass} = \text{base\_Mass} + \frac{\text{history\_Mass}}{2} + \frac{\text{Current\_Mass}}{2}. \quad (1)$$

For controlling the intensity of relevance between words, we use the concept of edge in physic to describe it.

(vi) Chain, it is an array for recording the ID of the backward relating word.

(vii) Max_flaw, it is the maximum associating strength regarding a group of coword.

(viii) Flaw, it describes the associating strength regarding a group of cowords.

(ix) Radius, it is the repulsion radius that keeps a minimum distance from other word-particles.

(x) Pull_Radius, it is the pulling radius, which means if other word-particles break in, they will be pushed away keeping the radius distance from the intruded word-particle.

(xi) Velocity, it defines a semantic shifting trend that can strengthen the relation of two force-affected words.

In Figure 1, the word-particles world and boy are backward related to hello, and the word-particle bye is an isolated word-particle. The intensity of relation between the word-particles hello and world is stronger than the intensity of relation between hello and boy. The directed line denotes the direction of word order, where the pointed word-particles can drive their linking word-particles. For example, the coword hello world appears more frequent than the coword hello boy.

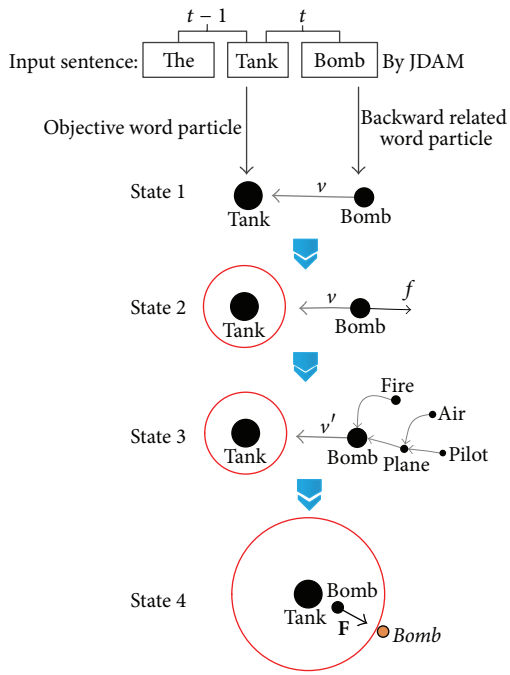


FIGURE 2: The semantic relating rule.

3.2. Semantic Relation Building. Based on the word-particle model, we define the semantic relating rule to control the motion of particles within a given text context. The documents for training play the role of a driven-force source, making the words have more opportunities to come together, which appear in similar contexts. The procedure is as follows.

Step 1. The word embedding is trained by the document. Each document will be sampled sentence by sentence via a 2-gram window. In the 2-gram window, the first word is assumed as the target object, and the second word is assumed as the associated object. The assumption means that the associated word-particle will be forced to move towards the target word-particle. The related word-particle will be given an impulse. This can drive the word-particle with a certain velocity. The progress is illustrated as state 1 in Figure 2. The word-particle bomb is associated with tank moving with the velocity v.

Step 2. Given an impulse, the word-particle can be initialized with a velocity. Meanwhile, it will be slowed down by the force that comes from friction, until its velocity reduces to zero and does not get in the repulsion radius of its objective word-particle. When the word-particle moves into the repulsion radius of the objective word-particle, it will be stopped at the edge keeping a distance of repulsion radius from the objective word-particle. This is shown as state 2. A velocity affects the word-particle tank, and the word-particle bomb is affected continuously by the friction force f.

Step 3. During a certain period of document learning, some word-particles will set up some relations with other word-particles. We establish a chain reacting rule for simulating the context feature. The rule specifies the impulsion transition in the way of particle by particle, and the initial energy will degrade at each reaction. This passive action simulates the phenomenon that a topic word in a document has more semantics and can be an index for document retrieval. The progress is controlled by (2). Given m_0 denotes the property Mass of the impacted word-particle and m_i denotes this property of other word-particles. The relation-building condition is

$$\begin{aligned} i &\in \text{Chain}, \\ d_i &> \text{Pull\_Radius}, \end{aligned} \tag{2}$$

where i denotes the ID of the ith word-particle that relates to the object word-particle, and d_i denotes the corresponding distance of the ith word-particle between the object word-particles. The velocity v_{t-1} will update v_t via (3), if the word-particle satisfies the condition. This procedure will repeat iteratively till the velocity falls to zero. For example, in state 3, the word-particle bomb has two backward associating particles fire and plane. Its initial velocity will be decomposed with plane according to (3), if given an impulsion towards tank. But the word-particle fire fails to move, because it is outside of the Pull_Radius distance of bomb. The decomposition will be delivered repetitively if the velocity fits the condition and is greater than zero.

$$m_0 v_{t-1} = \left(\sum_{i \in \text{Chain},\ d_i > \text{Pull\_Radius}}^{k} m_i + m_0 \right) v_t. \tag{3}$$

Step 4. We add a repulsion radius for keeping the uniqueness of every word-particle, because each word embedding is unique. When the moving word-particle intrudes the repulsion radius of other particles, it will stop and stay at the edge of the affected word-particles, keeping a repulsion radius distance. The progress is shown as state 4. Generally, the word relation network is expected growing stably; we present an inspecting criterion to check the convergence, which is as follows:

$$v_t = \frac{m_0 v_{t-1}}{\lim_{k \to \infty} \sum_{i \in \text{Chain},\ d_i > \text{Pull\_Radius}}^{k} m_i + m_0} \longrightarrow 0. \tag{4}$$

In (4), the initial velocity will trend to zero with the relation increasing of the number of associated word-particles; that is, $v_t \to 0$. When the movement of an activated particle becomes tiny, such convergence means the property Pos has reached relatively fixed coordinates. This indicates that the word already has situated in a relatively stable semantic network.

3.3. Semantic Relation Separating. For controlling the growth of words association, those words that are of low frequency in 2-gram context should be filtered, whereas those words with high frequency of relations should be retained. We propose to use a linear elastic fracture mechanics to control such filtering. A rope model is presented to represent the coword relation, which can be assumed as a flaw that comes from a type of material. The strengthening or weakening of a relation

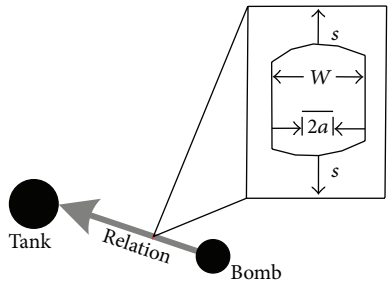


Figure 3: The flaw model.

between words is controlled via the corresponding flaw size. An illustration is shown in Figure 3.

More formally, given W denotes the width of rope, its value is obtained through the counting of a 2-gram sampling. Its maximum value corresponds to the property Max_flaw. Given $2a$ denotes the size of a flaw corresponding to the property Flaw. Given s is the pull force that is calculated by s = Mass × Velocity. We use the stress-intensity factor K to control the size of a flaw. K can be obtained as follows:

$$K = m_{\text{relation}} v \sqrt{\pi a} \frac{2a}{W}. \tag{5}$$

In (5), the variant m_{relation} corresponds to the property regarding the synthetically occurring frequency of a word-particle, and v corresponds to the velocity of an activated word-particle. The value of K is in proportion to the size of $2a$, which refers to the concept of flaw. Moreover, the flaw extending speed is denoted by da/dN as

$$\lg \frac{da}{dN} = \lg C + n \lg \Delta K. \tag{6}$$

In (6), $\lg C$ denotes a compensation constant, and n is a scale factor. da/dN is in proportion of K. The condition is that K will decrease if W goes beyond $2a$. When the size of flaw is up to W, a separation will happen in the semantic relation. This means the associated word-particles are no longer affected by the initial impulses, which are generated from their objective word-particles.

4. Simulations

In this section, we compare the proposed word embedding with the three classic NLM-based word embeddings, Huang 2012 [42], C&W [3], and M&S-Turian [20]. Huang 2012 is the word embedding that uses multiple embeddings per word. C&W is the embedding that is trained by a feedforward neural network, and M&S-Turian is the embedding that is obtained by an improved RBM training model. The two datasets Reuters 21578 and RCV1 are used for training and evaluation. In Reuters 21578, we extracted 21,578 documents from the raw XML format and discarded the original class labels and titles, only using the description section. The RCV1 contains 5,000 documents that are written from 50 authors. 70 percent of random sampling among these documents was used to train the word embedding, and the remainder was used to evaluate the NLP tasks of POS, NER, and SRL with other three types of embedding. All words will keep their original forms, whereas the numbers, symbols, and stop words are kept to be trained together. Those words that are not included in training corpus will be discarded. We regard these tasks as a classification problem and apply a unified feedforward neural linguistic model to perform the tasks. The compared word embeddings are readymade, but the parameters of the neural networks require to be trained by the corresponding embedding. We use the benchmark that is provided by [3]. The results regarding the NLP tasks are compared based on this benchmark.

4.1. Evaluation. The benchmark is measured in terms of precision, recall, and $F1$ [3, 20, 42]. Assuming N words are waiting for labeling, there exit $\{N1 \mid N1 \leq N\}$ words that are labeled correctly and $\{N2 \mid N2 \leq N\}$ words that are labeled wrong. The value of precision is used to evaluate the accuracy of labeling on POS, NER, and SRL. The precision can be obtained as

$$p = \frac{N_1}{N_1 + N_2}. \tag{7}$$

The value of recall is used to evaluate the coverage of labeling on POS, NER, and SRL. The recall can be calculated as

$$r = \frac{N_1}{N}. \tag{8}$$

The $F1$ is a combining evaluation with precision and recall, which are as follows:

$$F1 = \frac{2pr}{p + r}. \tag{9}$$

4.2. Training. The parameters of the physical system are set as follows. The coefficient of fiction is set to 0.1, the coefficient of gravity is set to 9.8, and the initial velocity is set to 2. The parameters that control semantic separating are set such that Max_flaw is set to 0.2, the initial value of Flaw is 0, Radius is set to 1, and Pull_Raduis is set to 20. For controlling the flaw extending speed, $\lg C$ is set to 1 and n is set to 1. We demonstrate the training result in terms of the generating procedure of word graph and average speed of word-particles. A word graph can give an intuitive visualization for observing a group of word relations. We test a small number of datasets to simulate the word embedding generation. The result is shown in Figure 4, which contains the names from 19 countries.

In Figure 4(a), all the words appear in the plain physical system and obtain a certain position, because the training documents had given some forces to direct the word arranging that follows the context in original text. But in this stage, some word-particles still have a certain degree of speed to move. Those frequent word-particles such as China, USA, Germany, and France have a relatively high speed; they move pulling their backward relating word-particles (Figures 4(b)–4(d)). For example, China has four backward relating word-particles, Pakistan, India, USA, and UK; Germany has two

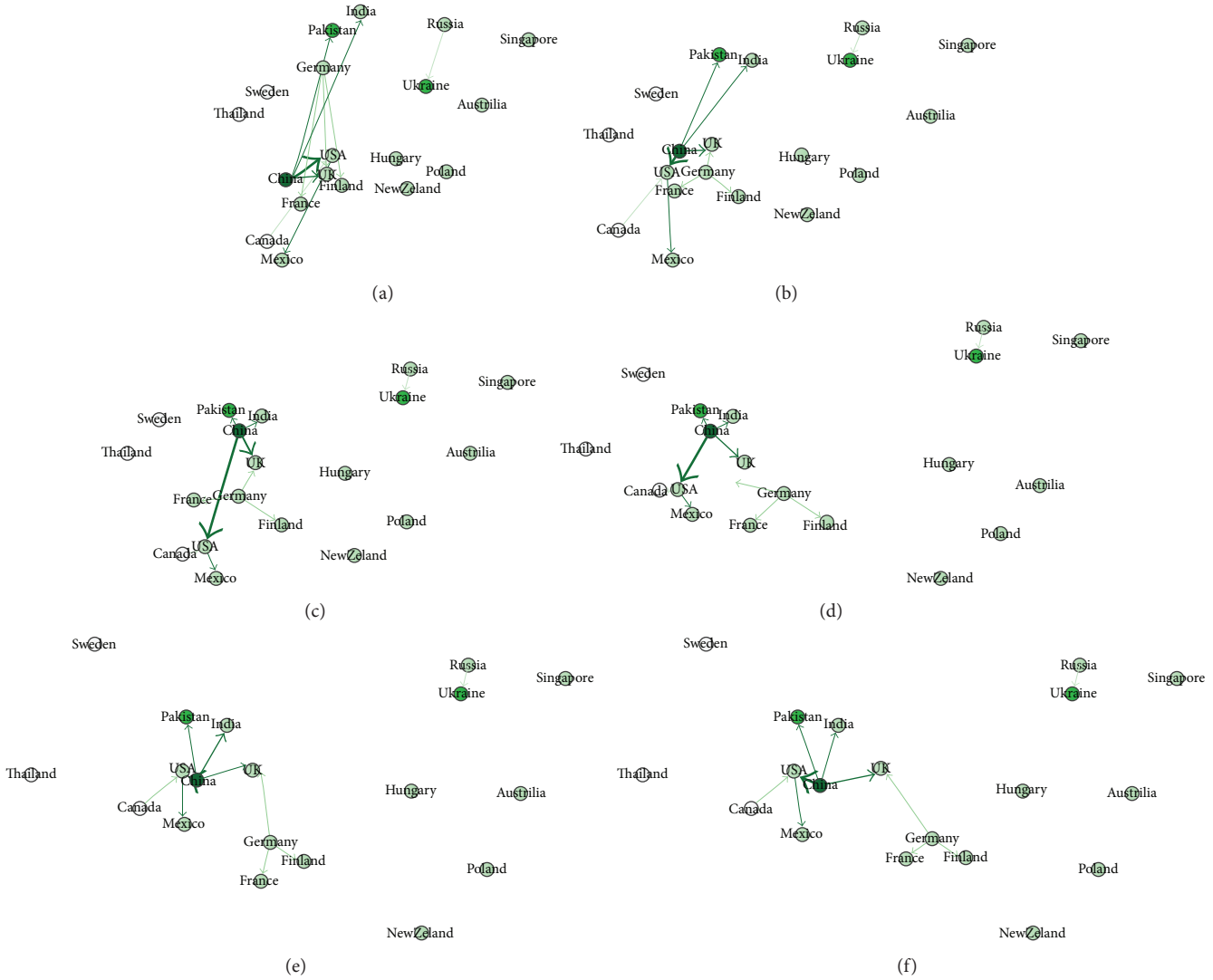


Figure 4: Force-directed embedding for country names.

backward relating word-particles, France and Finland. The other isolated word-particles have situated in a relatively stable position with a few movements. We can see that some word-particles overlay with each other (Figure 4(d)); for example, India and Pakistan are too close at China, and Canada overlays USA. The repulsing force starts to function at this time. The too close word-particles will push each other until they reach a balance distance (Figure 4(e)). When the inputting documents are all about similar topics, the positions of word-particles will also not vary too much, showing a relatively stable topological graph (Figure 4(f)).

The training result of dataset Reuters 21578 is shown in Figure 5. Each word-particle is colored with a green block. The intensity of the relation between word-particles is represented by a blue line, where the thicker lines mean a higher frequency of coword relation. The position of each word-particle is a 2-dimensional coordinate that is exactly the training result word embedding. The result shows that the numbers of word relations and new word-particles will grow with the training, which iterates from 1,000 to 10,000. The particles system expands the region outward gradually. The particle distribution presents as an ellipse for accommodating more new words-particles.

The training result of RCV1 is shown in Figure 6. Such distance-based semantic measurement can be interpreted from some viewpoints. For example, the country and geographical word-particles, German, Russian, U.S., and Beijing, are clustered together. The geo-related word-particles, Niagara, WTO, and U.S.-based, are pushed closely to these words. Such word-particle graph can present an intuitive result for evaluating the training of word embedding; no matter during the training or after training, we can intervene the position of word-particle to improve the final result in a data-visualization based way.

On the other hand, we use (3) to estimate the average velocity of word-particles for evaluating the training process from a statistics viewpoint. When the velocity decreases to 1 below, the convergence is assumed to be happening

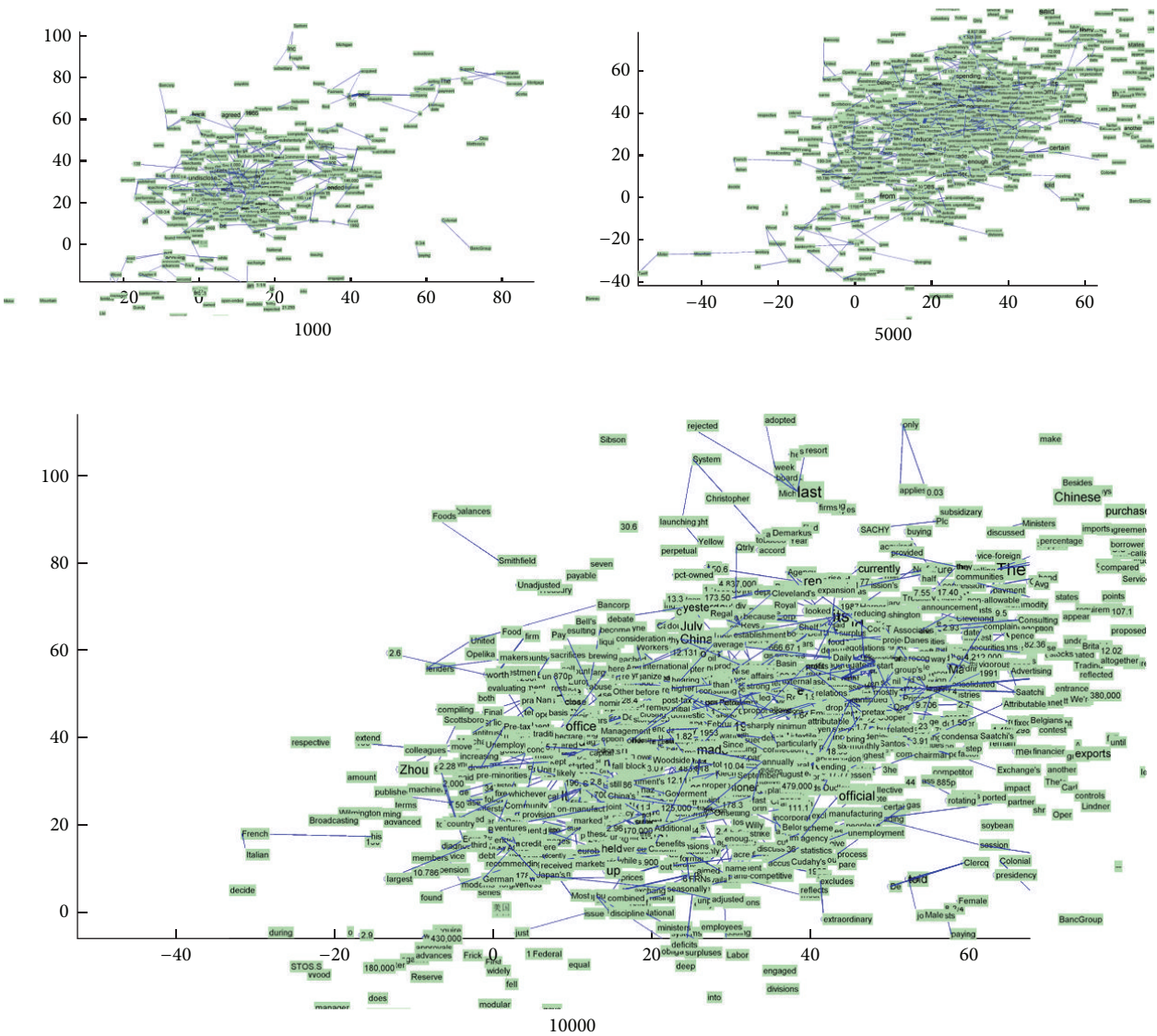


Figure 5: Force-directed embedding for Reuters 21578.

and the assumption coincides with the reality roughly. We experiment 50 times for the two datasets, respectively. The result is shown as the boxplots in Figure 7. From the downwards trends of average velocity, the assumption that a word-semantic network will be stabilized in a certain number of similar documents coincides with the result of two datasets roughly. Both the convergences of the two datasets' training appear at the training stage around the 20,000th documents.

In the presented word embedding learning, there is not a specific converging criterion for terminating the training, because the object functions of these neural based models are nonconvex so there may not exist an optimum value theoretically. Empirically, these word embedding learning methods require repeating documents 20 or 30 times [3]. It brings a serious problem that time consumption is proportional to the number of documents. Such procedure usually requires undergoing a long-term training time [3, 30]. But in our proposed model, we demonstrate that setting a semantic convergence condition is more convenient to select than those neural based approaches. The convergence criterion provides a more explicit direction for word embedding learning. Meanwhile, the result demonstrates that to learn an acceptable word embedding requiring a certain number of documents, small or medium scale datasets may not be appropriate.

5. Reuters 21578 and RCV1

For testing the usability, we compare the word embedding with other three word embeddings on the NLP tasks, using the same type of neural linguistic model. The testing items are performed on POS, NER, and SRL. The results are listed in Tables 2 and 3. In Reuters 21578, the labeling system using our word embedding obtains 91.0% on POS, 83.4% on NER, and 67.3% on SRL for $F1$. This $F1$ score gets the third place in POS and the second place on both NER and SRL. In RCV1, it

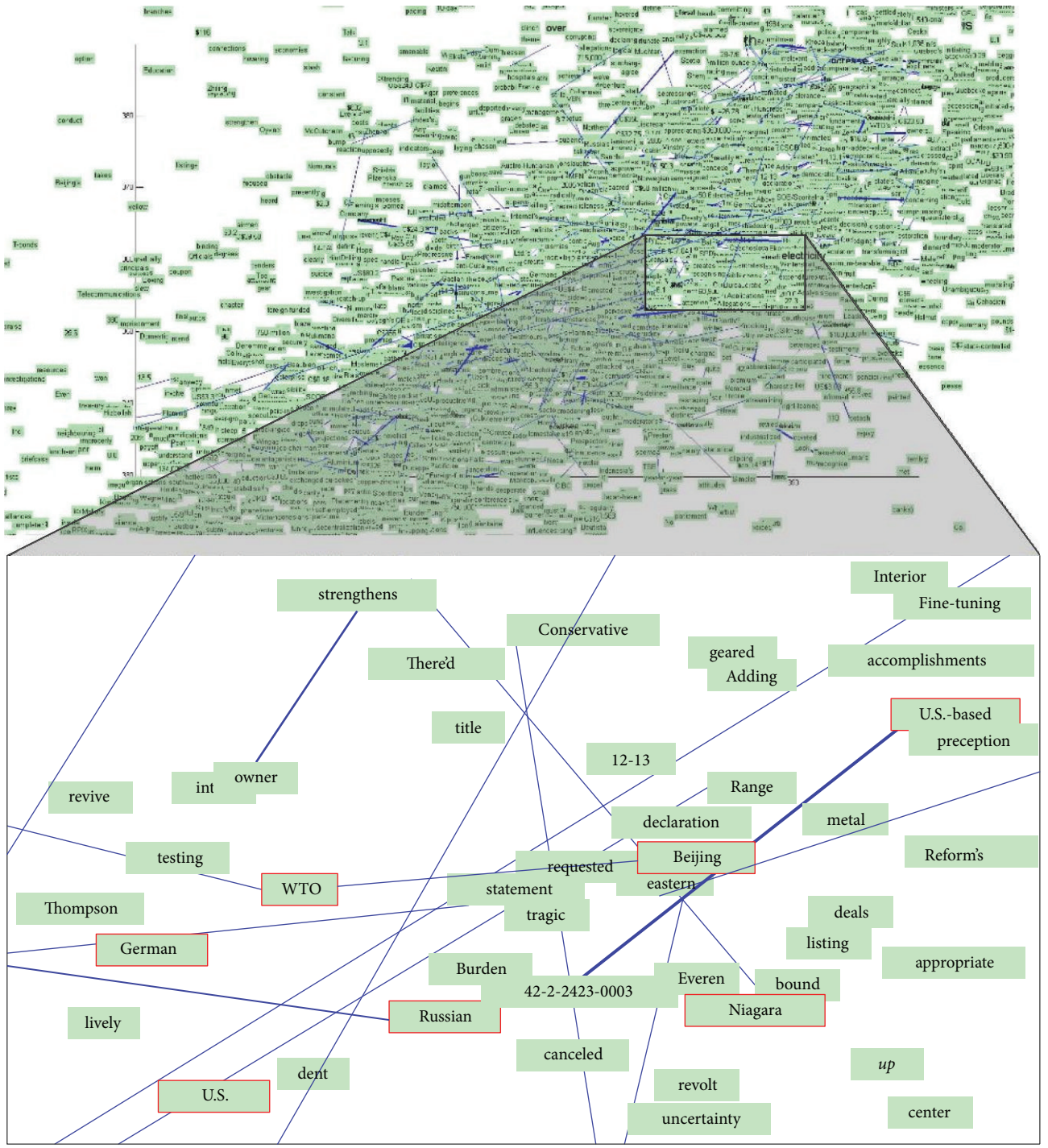


Figure 6: Force-directed embedding for RCV1.

achieves 89.1% on POS, 82.1% on NER, and 65.9% on SRL for $F1$. The $F1$ scores obtain the second place in POS, the third place in NER, and the second place in SRL.

The performance of the proposed word embedding is close to the best results in [3], but the dimensional number is two, which is far less than the 50- or 100-dimensional word embeddings [3, 43]. This brings a benefit for reducing the number of neural cells in performing NLP tasks by such type of linguistic models. Implementing these NLP tasks, we construct a 3-layer feedforward neural network with a 5-cell inputting layer and a 100-cell middle layer and a 25-cell outputting layer. To utilize the compared word embeddings, the number of the inputting vectors is set to 500, because all of them are the 100-dimensional embeddings. But our corresponding inputting layer just requires 10-dimensional vector. The structure of model is simplified

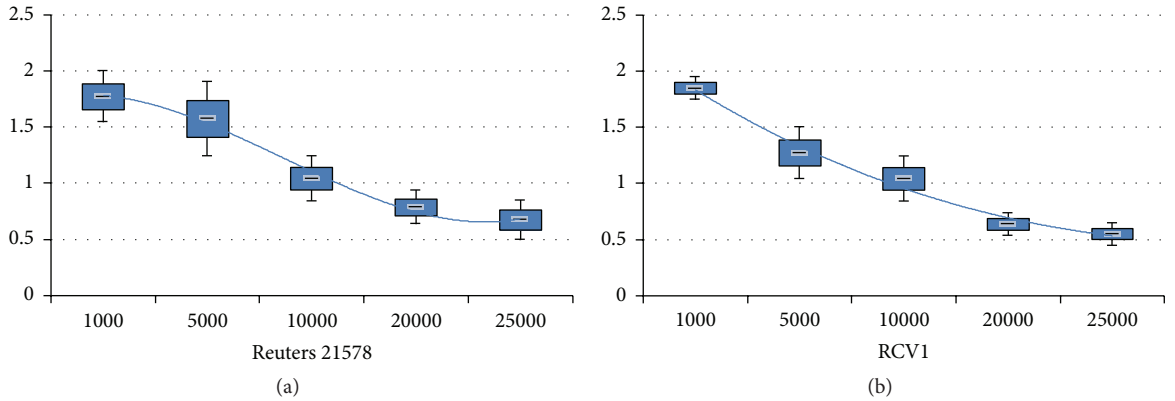


Figure 7: The y-axis represents the velocity of a word-particle; the x-axis represents the number of documents. Both of them show the downwards trends.

Table 2: Comparison of POS/NER/SRL on Reuters 21578.

	Precision	Recall	$F1$
Force-directed	*92.5/84.1/70.2*	*89.5/82.7/64.6*	*91.0/83.4/67.3*
Huang 2012	94.2/86.2/74.8	93.8/86.8/74.1	94.0/86.5/74.4
C&W	93.6/82.4/67.8	92.8/81.5/65.8	93.2/81.9/66.8
M&S-Turian	91.4/82.1/63.6	86.2/75.2/62.5	88.7/78.5/63.0

Table 3: Comparison of POS/NER/SRL on RCV1.

	Precision	Recall	$F1$
Force-directed	*90.1/82.7/66.3*	*88.2/81.5/65.5*	*89.1/82.1/65.9*
Huang 2012	88.4/83.2/71.3	90.7/84.6/70.3	89.5/83.9/70.8
C&W	88.5/83.5/64.6	85.2/82.6/63.1	86.8/83.0/63.8
M&S-Turian	90.8/80.5/63.6	90.3/73.9/65.7	90.5/77.1/64.6

which can reduce the complexity of neural networks. This advantage will improve the performance of such models, such as reducing training time and improving the speed of labeling. The result also demonstrates that learning a group of word embeddings cannot be high dimensional and depend on the neural network based approaches. It means word representation learning and the task system constructing can be decomposed to two individual parts. The two-step framework could achieve the same goal with the all-in-one models [3].

6. Conclusions and Future Work

In this paper, we propose a force-directed method that uses a fracture mechanic model to learn word embedding. The result demonstrates that the physical simulation approach is feasible. It improves the procedure of the traditional NLM-based approaches in terms of parameters training and tasks performing (POS and NER and SRL). The next works are as follows. The model will be improved to suit streaming data: using a one-step solution for predicting the coordinate of word-particle, which will improve the performance of our system; packaging the properties of word-particle with the .gefx file format (Graph Exchange XML Format) that can provide a capability for data sharing across multiple data visualizing tools, for example, Gephi.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this manuscript.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[2] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '15)*, 2015.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 3, no. 12, pp. 2493–2537, 2011.

[4] J. Turian, L. Ratinov, Y. Bengio, and D. Roth, "A preliminary evaluation of word representations for named-entity recognition," in *Proceedings of the NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, 2009.

[5] S. R. Bowman, C. Potts, and C. D. Manning, "Learning distributed word representations for natural logic reasoning," in *Proceedings of the AAAI Spring Symposium on Knowledge Representation and Reasoning*, Stanford, Calif, USA, March 2015.

[6] B. Fortuna, M. Grobelnik, and D. Mladenić, "Visualization of text document corpus," *Informatica*, vol. 29, no. 4, pp. 497–502, 2005.

[7] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of the International Workshop on Artificial Intelligence and Statistics (AISTATS '05)*, vol. 5, pp. 246–252, Barbados, Caribbean, 2005.

[8] R. Navigli and S. P. Ponzetto, "BabelNet: building a very large multilingual semantic network," in *Proceedings of the 48th*

Annual Meeting of the Association for Computational Linguistics (ACL '10), pp. 216–225, 2010.

[9] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "Learning distributed word representations for bidirectional LSTM recurrent neural network," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL '16)*, San Diego, Calif, USA, June 2016.

[10] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 641–648, June 2007.

[11] Z. Li, H. Zhao, C. Pang, L. Wang, and H. Wang, *A Constituent Syntactic Parse Tree Based Discourse Parser*, CoNLL-2016, Shared Task, Berlin, Germany, 2016.

[12] Z. Zhang, H. Zhao, and L. Qin, "Probabilistic graph-based dependency parsing with convolutional neural network," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL '16)*, pp. 1382–1392, Association for Computational Linguistics, Berlin, Germany, August 2016.

[13] R. Wang, M. Utiyama, I. Goto, E. Sumita, H. Zhao, and B.-L. Lu, "Converting continuous-space language models into N-gram language models with efficient bilingual pruning for statistical machine translation," *ACM Transactions on Asian Low-Resource Language Information Process*, vol. 15, no. 3, pp. 1–26, 2016.

[14] G. Mesnil, A. Bordes, J. Weston, G. Chechik, and Y. Bengio, "Learning semantic representations of objects and their parts," *Machine Learning*, vol. 94, no. 2, pp. 281–301, 2014.

[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[16] G. Salton, A. Wong, and C. S. Yang, "Vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[17] E. D'hondt, S. Verberne, C. Koster, and L. Boves, "Text representations for patent classification," *Computational Linguistics*, vol. 39, no. 3, pp. 755–775, 2013.

[18] C. Blake and W. Pratt, "Better rules, fewer features: a semantic approach to selecting features from text," in *Proceedings of the IEEE International Conference on Data Mining (ICDM '01)*, pp. 59–66, IEEE, San Jose, Calif, USA, 2001.

[19] M. Mitra, C. Buckley, A. Singhal, and C. Cardie, "An analysis of statistical and syntactic phrases," in *Proceedings of the 5th International Conference Computer-Assisted Information Searching on Internet (RIAO '97)*, pp. 200–214, Montreal, Canada, 1997.

[20] 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 Association for Computational Linguistics (ACL '10)*, pp. 384–394, July 2010.

[21] M. Sahlgren, "Vector-based semantic analysis: representing word meanings based on random labels," in *Proceedings of the Semantic Knowledge Acquisition and Categorisation Workshop at European Summer School in Logic, Language and Information (ESSLLI XIII '01)*, Helsinki, Finland, August 2001.

[22] M. Sahlgren, *The Word-Space Model: Using Distributional Analysis to Represent Syntagmatic and Paradigmatic Relations between Words in High-Dimensional Vector Spaces*, Stockholm University, Stockholm, Sweden, 2006.

[23] P. Cimiano, A. Hotho, and S. Staab, "Learning concept hierarchies from text corpora using formal concept analysis," *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 305–339, 2005.

[24] A. Kehagias, V. Petridis, V. G. Kaburlasos, and P. Fragkou, "A comparison of word- and sense-based text categorization using several classification algorithms," *Journal of Intelligent Information Systems*, vol. 21, no. 3, pp. 227–247, 2003.

[25] M. Rajman and R. Besancon, "Stochastic distributional models for textual information retrieval," in *Proceedings of the 9th Conference of the Applied Stochastic Models and Data Analysis (ASMDA '99)*, pp. 80–85, 1999.

[26] G. E. Hinton, "Learning distributed representations of concepts," in *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pp. 1–12, 1986.

[27] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biological Cybernetics*, vol. 61, no. 4, pp. 241–254, 1989.

[28] T. Honkela, V. Pulkki, and T. Kohonen, "Contextual relations of words in grimm tales, analyzed by self-organizing map," in *Proceedings of the Hybrid Neural Systems*, 1995.

[29] T. Honkela, "Self-organizing maps of words for natural language processing application," in *Proceedings of the International ICSC Symposium on Soft Computing*, 1997.

[30] T. Mikolov, S. W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '13)*, 2013.

[31] W. Xu and A. Rudnicky, "Can artificial neural networks learn language models," in *Proceedings of the International Conference on Statistical Language Processing*, pp. 1–13, 2000.

[32] M. I. Mandel, R. Pascanu, D. Eck et al., "Contextual tag inference," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 7S, no. 1, 2011.

[33] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," *Journal of Machine Learning Research*, vol. 22, pp. 127–135, 2012.

[34] F. Huang, A. Ahuja, D. Downey, Y. Yang, Y. Guo, and A. Yates, "Learning representations for weakly supervised natural language processing tasks," *Computational Linguistics*, vol. 40, no. 1, pp. 85–120, 2014.

[35] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[36] A. Mnih and G. Hinton, "A scalable hierarchical distributed language model," in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS '08)*, pp. 1081–1088, December 2008.

[37] G. Mesnil, Y. Dauphin, X. Glorot et al., "Unsupervised and transfer learning challenge: a deep learning approach," *Journal of Machine Learning Research*, vol. 27, no. 1, pp. 97–110, 2012.

[38] S. G. Kobourov, "Spring embedders and force directed graph drawing algorithms," in *Proceedings of the ACM Symposium on Computational Geometry*, Chapel Hill, NC, USA, June 2012.

[39] M. J. Bannister, D. Eppstein, M. T. Goodrich, and L. Trott, "Force-directed graph drawing using social gravity and scaling," in *Graph Drawing*, W. Didimo and M. Patrignani, Eds., vol. 7704 of *Lecture Notes in Computer Science*, pp. 414–425, 2013.

[40] A. Efrat, D. Forrester, A. Iyer, S. G. Kobourov, C. Erten, and O. Kilic, "Force-directed approaches to sensor localization," *ACM Transactions on Sensor Networks*, vol. 7, no. 3, article 27, 2010.

[41] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proceedings of the*

International Symposium on Physical Design (ISPD '05), pp. 185–192, Santa Rosa, Calif, USA, April 2005.

[42] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multipleword prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*, pp. 873–882, July 2012.

[43] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the INTERSPEECH*, 2010.

Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 9821608, 11 pages
http://dx.doi.org/10.1155/2016/9821608

Research Article

A Character Level Based and Word Level Based Approach for Chinese-Vietnamese Machine Translation

Phuoc Tran,[1] Dien Dinh,[2] and Hien T. Nguyen[1]

[1]*Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam*
[2]*Faculty of Information Technology, VNU-HCM University of Science, Ho Chi Minh City 700000, Vietnam*

Correspondence should be addressed to Dien Dinh; ddien@fit.hcmus.edu.vn and Hien T. Nguyen; hien@tdt.edu.vn

Received 11 March 2016; Accepted 8 May 2016

Academic Editor: Stefan Haufe

Copyright © 2016 Phuoc Tran et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Chinese and Vietnamese have the same isolated language; that is, the words are not delimited by spaces. In machine translation, word segmentation is often done first when translating from Chinese or Vietnamese into different languages (typically English) and vice versa. However, it is a matter for consideration that words may or may not be segmented when translating between two languages in which spaces are not used between words, such as Chinese and Vietnamese. Since Chinese-Vietnamese is a low-resource language pair, the sparse data problem is evident in the translation system of this language pair. Therefore, while translating, whether it should be segmented or not becomes more important. In this paper, we propose a new method for translating Chinese to Vietnamese based on a combination of the advantages of character level and word level translation. In addition, a hybrid approach that combines statistics and rules is used to translate on the word level. And at the character level, a statistical translation is used. The experimental results showed that our method improved the performance of machine translation over that of character or word level translation.

1. Introduction

Unlike Western languages, typically English, the words in Chinese and Vietnamese are not distinguished by spaces [1]. A Chinese sentence includes a series of characters, including punctuation, and they are located next to each other without any spaces. In Vietnamese, spelling words are separated by one space, and the punctuation is located immediately after the spelling words. Therefore, the word segmentation (WS) problem often is addressed first in natural language processing in general and in machine translation (MT) from Chinese or Vietnamese into other languages in particular and vice versa.

When translating the language pairs in which the words are not segmented by spaces, WS should be taken into consideration. Based on our experiments, we found that segmenting and not segmenting a word have their own advantages and disadvantages. The most obvious advantage of the word unsegmentation machine translation system (also called character segmentation) is that it is difficult to generate new words, but its disadvantage is that it obviously mistranslates some words. (The Chinese words in Figure 1 are some examples.) These cases are usually the translation of words in which the words have meanings that do not relate to the characters that created them, especially the Sino-Vietnamese (SINO) and named entity (NE). On the contrary, the WS machine translation system gives better translations, but it generates many unknown words (UKWs). Furthermore, in the low-resource language pairs, such as Chinese and Vietnamese, the issue of generating UKWs is worse. Figure 1 shows the errors in character level (CL) translation systems and word level (WL) translation systems in our testing corpus.

The word "王红" ("Vuong Hong") is a Chinese person name and must be translated into Sino-Vietnamese as "Vương Hồng" [2], but in CL translation systems, the word "红" is translated into "đỏ" (red). Similarly, the word "出色" ("remarkable") is translated into "xuất sắc" (Sino-Vietnamese word), but the CL translation systems give the translation "ra màu" (出/"ra" ("to go out," "to come out") và 色/"màu" ("color")). In the remaining case, the digit string "一一九" is translated into "một 9" in the CL translation systems, while it

Chinese sentence	王红	出色	一一九
Character based translation	Vương đỏ	ra màu	một 9
Word based translation	王红	出色	一一九
Reference translation	Vương Hồng	xuất sắc	119

Figure 1: Examples for incorrect translations of CL and WL translation systems.

should be "119." The WL translation systems fail in these three cases.

Obviously, for language pairs in which the words' boundaries are not clear, such as Chinese and Vietnamese, the choice of CL or WL translation systems has its own flaws. In this paper, we propose a Chinese-Vietnamese translation model based on the combination of CL and WL translation models. At the word level, we construct the model in two STEPs. Step 1 is to use WL statistical machine translation (WL-SMT), and Step 2 is to use rule-based MT to translate SINO and NE words, which were not translated in Step 1. In the CL model, the words that are not translated by the WL model will decay into characters and be translated based on statistics.

The rest of the paper is structured as follows: Section 2 introduces some related works. Section 3 gives the background about Sino-Vietnamese translation and NE translation. Section 4 provides a detailed description of our proposed model. Section 5 shows and discusses the results of our experiments. Section 6 summarizes our work and gives our main conclusions.

2. Related Work

In this paper, we focused on surveying WS methodologies to increase the performance of SMT. The term "segmented word" does not quite agree with linguists' concept of a word. A word here may be a pragmatic word (greater than the linguist's word) or a morpheme (smaller than the linguist's word). The purpose of adjusting WS is how the words in the training corpus are covered as much as possible, and the word alignment results have more 1-1 mappings.

Depending on the characteristics of translated language pairs, their WS approaches are different. To the best of our knowledge, there are some major WS approaches for MT, including morphology-based WS; anchor language-based (AL-based) WS; a combination of character segmentation and word segmentation; and other approaches.

2.1. Word Segmentation Based on Word Morphology. Lee [3] presented a morphological analysis algorithm for making morphologically and syntactically symmetrical language pairs with different structures. The algorithm segmented words in the rich morphological language in the form, prefix (es)- stem-suffix (es), and tagged the part of speech for bilingual corpus. Then, the algorithm detects the morphemes and decides whether to merge or delete the morphemes in the rich morphology language; the aim of this work was to make this language pair morphologically and structurally symmetrical.

In addition, in [4], Goldwater and McClosky used morphological analysis to reduce the sparse data problem in MT and increase the similarity between the two languages, thus improving the quality of machine translation for a highly inflected language, such as Czech in Czech-English translations.

2.2. Word Segmetation Based on the Anchor Language. In this approach, many authors based their work on one of two languages in MT to standardize WS in the other language. Chinese-English or English-Chinese translation are methods that use this approach. By using the word boundary in English, the authors conducted segment words based on the word alignments between English words and Chinese characters.

Specifically, in [5], the authors used English words as the anchor to segment the Chinese words. In this approach, the Chinese characters are combined into a word if they are aligned with the same English word. Moreover, because English NE (name of a person, name of an organization, and the name of a location) are capitalized, the Chinese words can be segmented basing on these English NEs to have better results (Chinese is case-insensitive). However, this method has a disadvantage as well as the common disadvantage in learning from bilingual corpora, which have sparse data (even though the corpora are very large) and noisy data. This leads to problems in that the system is not able to segment or it segments incorrectly the obvious words in the dictionary. Also, in this paper, the authors observed the reduction of the quality of Chinese-English machine translation when the Chinese words were not segmented.

By developing the same idea, Ma and Way [6] proposed the WS method using the results of word alignment between Chinese characters and English words. For word alignment results, the authors extracted the candidate words (n – 1 alignments between Chinese and English). Then, they selected the candidate words by the cooccurrence frequency of the characters inside that candidate's word.

In [7], Paul et al. expanded the WS for any language pairs for which the source language is unsegmented and the target language segmentation is determined, such as Chinese-English or Japanese-English. Furthermore, the authors proposed to decode directly from unsegmented text. They used the WS information in a phrase table to segment the input text. This avoided the inconsistency of WS between the input sentence and the phrase table.

Decomposition of Chinese words into smaller meaningful morphemes also is a popular method for improving the quality of the machine translation [8]. In this approach, first, the authors segmented the Chinese words. Then, they used the Chinese-English word alignments to filter the $1 - n$ alignments and adjust the alignments. Chinese with polysyllabic words, including more than one meaningful morpheme word, is translated into English words. For example, "教育署" is translated into "Department of Education" in English ("署" means "Department," and "教育" means "Education"). This WS approach reduces the total cooccurrence of Chinese-English word pairs and gives more $1 - n$ alignment. For

example, because "教育署" is a word, it does not contribute anything for the "教育/Education" pair and the "署/Department" pair, whereas the "教育署/Department of Education" creates the $1 - n$ alignments, such as "教育署 → Education" and "教育署 → Department." Therefore, the authors proposed this method to decompose Chinese words into smaller, meaningful morphemes.

Similarly, Wang et al. [9] improved WS by using the results of manually aligning words from the bilingual corpus. A new point of the paper is to propose the concept of "atomic block" in the English language. The "atomic block" is the English words plus the compounds, for example, "rely on" or "carry out." A Chinese word should be decomposed into smaller words if it aligns with any English "atomic block" (or $1 - n$ alignment).

In [10], Chu et al. segmented words for Chinese-Japanese language pairs. Both of them required WS. However, the Japanese segmenter toolkit has an accuracy up to 99%, and it is not necessary to correct WS. Japanese is used as the AL to segment Chinese. Moreover, the system also has used the shared Chinese characters (Kanji) in Japanese in order to increase the quality of Chinese WS. This work was rather close to ours, but there were some differences in the following points: (1) we translate Chinese sentences based on the combination of characters and the word level; (2) the linguistic relationships we used were NE and SINO. They are not the same as Kanji in Japanese (they are outlined in Sections 3.1 and 3.2).

According to this WS method, the unsegmented language (UL) is segmented and then, based on the word alignment results of UL and AL, the system will decay UL's words into smaller units. For example, in [8], the Chinese words that were aligned with many English words were decomposed. For example, the "洗衣机/washing machine" word pair can be decomposed into two word pairs, that is, "洗衣/washing" and "机/machine," but the "暖气机/heater" word pair cannot be decomposed into "暖气" and "机," because the word is only aligned with a single English word, that is, "heater."

2.3. Combination of Character Segmentation and Word Segmentation. Xu et al. [11] proposed a method to perform different ways of word WS and present them as a lattice. The input sentence of the translation system is a lattice that includes different ways of WS (not a sentence in which words are segmented), and the selection of a suitable WS is performed only at the translation step. The main idea of this approach is a two-stage method, that is, from character strings to word strings and from word strings of the source language to word strings of target language.

To date, there has been only one study related to our method and that was the approach that was proposed by Zhao et al. [12]. To the best of our knowledge, this is the first approach in which Vietnamese is translated into Chinese. In their approach, they performed the translation in two steps as follows.

Step 1. Using a bilingual dictionary to find Chinese characters that correspond to Vietnamese syllables: The system performs a maximum matching algorithm on the Chinese-Vietnamese dictionary to segment the Vietnamese words. The system uses two bilingual dictionaries to segment the Vietnamese words and to convert them into Chinese characters. The first dictionary is the Sino-Vietnamese dictionary. For Sino-Vietnamese words (that appear in the dictionary), the translation process is simple, and it can be used to look up words without any ambiguities. For the words that are not Sino-Vietnamese words, the system translates them by using a phrase table. The translation system automatically collects Vietnamese monolingual corpora from the Internet. Then, these corpora are translated into Chinese by using Google Translate. After that, the system has Vietnamese-Chinese bilingual corpora. (Vietnamese is single syllables, and Chinese is characters.) Following that, the system uses GIZA++ to align words and create a phrase table. From this table, the system takes out the aligned phrases on the condition that these phrases have the same number of syllables-characters. In the case in which a Vietnamese phrase is aligned with more than one Chinese phrase, the system will choose the phrase that has the highest probability of being the correct match.

Step 2. Using monolingual Chinese to modify and change the order of words, because the order of Vietnamese words is different from the order of Chinese words. The word order must be changed to obtain the correct order in Chinese. However, Vietnamese words do not always have only one Chinese meaning. To improve this, the system replaces Chinese words (translated by Vietnamese words) by more suitable Chinese words. A synonym dictionary is used to list all possible words. The system uses a language model to change word order and to determine the most possible words to replace.

The paper also indicates some limits:

(i) The Sino-Vietnamese dictionary limits only two words.

(ii) Most of the errors appear in phrase table.

(iii) The word order after the modification is not good.

Our approach also uses Sino-Vietnamese words as a factor in the Chinese-Vietnamese translation process. However, our approach is different from the other one, as indicated below:

(i) Our approach segments Chinese words and Vietnamese words by using corpora. The main advantage of this method is that it can recognize NEs that are not covered by dictionaries.

(ii) Our phrase tables, which were created by the Computational Linguistics Center (http://www.clc.hcmus.edu.vn/) (University of Sciences, HCMC-VNU), have better quality than this work's phrase tables, which were created from the bilingual corpora collected from Google Translate.

(iii) Translation at the word level includes two stages in our system, that is, (1) translation using word level and (2) translation of SINO and NEs. In stage (1),

if the words that have SINO or NE forms exist in the phrase table, they will be translated statistically. For Sino or NE forms that do not exist in the phrase table, they will be translated by rules. There are two reasons for this; that is, (1) there are some SINO words that are used less than their pure Vietnamese words, so our system will choose the meaning for the SINO words if they exist in the phrase table and (2) some foreign NEs are transliterated into Chinese (especially people's names). These kinds of NEs will have incorrect translations if they are translated by the rules of the system.

(iv) In the decoding step, we combine a statistical decoding model and a rule-based decoding model. The purpose of this strategy is to take advantage of the translation capability and word reordering capability of SMT models and the accuracy of translation by using the rules for NEs and SINO.

2.4. Other Methods. In this section, we present some WS methods that are different from the three methods discussed above. In [13], Zhao et al. assumed that the WS depended on the translation direction. The optimal WS is not sure to bring a better-quality translation. These authors argued that the corpus or dictionary used in WS will affect MT, so they must be optimized.

A different approach to the WS problem is based on unsupervised or semisupervised learning. In the work of Xu et al. [14], a Bayesian semisupervised learning model was proposed to segment Chinese words. They used monolingual and bilingual information to create WS suitable for SMT. Particularly, in the work of Nguyen et al. [15], an unsupervised WS model was used for MT. This model combined monolingual segmentation techniques and the bilingual word alignment model to adjust WS of the source sentence.

An extended version of this method was performed by Wang et al. [16]. The authors used the unsupervised learning method for segmenting Chinese words on large-scale corpora to support SMT. In addition, the authors modeled bilingual, unsupervised WS based on monolingual, unsupervised WS to improve the efficiency of WS, and they replaced Gibbs sampling with expectation maximization in the training process. This is considered to be the first work that used the bilingual, unsupervised WS method for large corpora. Furthermore, Zeng et al. [17] recommended using knowledge constraints to guide the monolingual supervised WS model. The authors also used "Chinese character-English word alignment" to extract word boundary distribution for character trigrams.

Like the previous approaches, these authors' methods are also applied to the language pairs in which the source language is not segmented and the word boundary of the target language is known (such as Arabic-English and Chinese-English).

3. Sino-Vietnamese and Named Entity

3.1. Sino-Vietnamese. Many Vietnamese words are borrowed from Chinese (normally called Sino-Vietnamese, which makes up about 65% of all Vietnamese words) [2]. Chinese, even in China, is pronounced differently, depending on the area, because there are many different voices or pronunciations, such as Cantonese, Hokkien, and Madarin. Some neighboring countries of China have their own reading of Chinese, such as Korea's having Sino-Korean (汉朝), Japan's having Sino-Japanese (汉和), and Vietnam's having Sino-Vietnamese (汉越). Thus, Sino-Vietnamese is the way Vietnamese people read. For example, the Chinese word "银行" is read as "yin hang" in Chinese and as "ngân hàng" in Vietnamese. A Chinese character can be pronounced as many Sino-Vietnamese words but, in a specific context, a Chinese character only corresponds to a specific Sino-Vietnamese word. As the above example "银行," the Sino-Vietnamese word for "银" is "ngân," and "行" is "hành," "hạnh," "hàng," or "hạng"; but when "银" and "行" are combined into a word, this word is only pronounced "ngân hàng."

Most Chinese-Vietnamese words that have Sino-Vietnamese pronunciations are Sino-Vietnamese words (called "standard Sino-Vietnamese words" or "pure Sino-Vietnamese words"). The words "văn hóa" (文化) and "hiện tại" (现在) are good examples. However, there are some Sino-Vietnamese words that have different meanings to the Vietnamese people compared to orthodox Chinese. For example, in Chinese, the word "博士" (Sino-Vietnamese is "bác sĩ" (physician in Vietnamese)) is used for doctorate and "bác sĩ" (physician in Chinese) is called "y sinh" (医生) or "đại phu" (大夫).

A Sino-Vietnamese word is the smallest meaningful unit. If we split it to smaller parts, these parts will be meaningless or have different meanings. Therefore, in Chinese-English machine translation, we do not split the Sino-Vietnamese words into smaller parts.

3.2. Named Entity in Chinese and Vietnamese. In this paper, we divide NEs into four categories, that is, (1) person name (PER), (2) organization name (ORG), (3) location name (LOC), and (4) number expression (NumExp) (date, time, percentage, number, and phone number). The Chinese words that belong to (1), (2), and (3) usually are translated into Vietnamese by their Sino-Vietnamese transliterations. NumExp is translated into Vietnamese by using grammatical transformation. Words in NumExp include digits combined with keywords that represent each kind of NumExp.

Like Vietnamese PER, Chinese PER are formed under the following structure: ⟨family name⟩⟨given name⟩, where both ⟨family name⟩ and ⟨given name⟩ have the length of one or two characters. For example, in the PER "赵经生" (Triệu Kinh Sinh), "赵" is a ⟨family name⟩ and "经生" is a ⟨given name⟩. In addition, to express a close relationship with elderly people, Chinese people usually use the word "老" (Lão: old) before ⟨family name⟩. For example, "老张" (Lão Trương) is used to refer to an elderly person who has the ⟨family name⟩ "Trương." Young people are addressed in a similar manner; Chinese people usually use "小" (Tiểu: small) before ⟨family name⟩ to express the close relationship. For example, the word "小王" is used to call a child who has the ⟨family name⟩ "Vương." This way of calling names usually omits the ⟨given name⟩ part in Chinese PER structures.

Table 1: Chinese numeric characters (from 0 to 9).

Chinese numbers	零	一	二	三	四	五	六	七	八	九
Vietnamese numbers	0	1	2	3	4	5	6	7	8	9

Table 2: Chinese unit characters.

Chinese unit characters	十	百	千	万	亿
Vietnamese numbers	10	100	1,000	10,000	100,000,000

For Chinese LOC, the length of an LOC does not exceed 10 characters, and an LOC follows the structure ⟨location name⟩⟨keyword⟩. In this structure, ⟨location name⟩ is a word item in the list of Chinese location names (about 30,000 location names). It usually ends by a keyword (about 120 keywords). For example, in the word "北京市" (Beijing City), "北京" is a ⟨location name⟩, and "市" is a ⟨keyword⟩.

ORG is more complicated than person names or location names because an ORG usually includes the combination of different entities. The maximum length of an ORG is normally 15 characters. For example, in the ORG "北京语言学院" ("Học Viện Ngôn Ngữ Bắc Kinh": "Beijing Languages Institute"), 北京 (Bắc Kinh) is an LOC, and 学院 is a ⟨keyword⟩.

NumExp includes some different types, such as numbers, phone numbers, order numbers, fractions, decimal numbers, dates, and times. Words in NumExp include digits combined with the keyword representing each type of NumExp. Therefore, number has an important role in constructing Chinese NumExp.

Like Vietnamese, Chinese numbers also are formed from the combination of characters that are similar to the numbers 0 to 9 in Vietnamese. The number of Chinese characters is presented in Tables 1 and 2.

A small difference between Chinese and Vietnamese is that numbers, such as 100, 1,000, 10,000, and 100,000,000, have their own words in Chinese (called unit characters). For example, the number "四百三十" is translated into "430" in Vietnamese.

4. Chinese-Vietnamese Translation Based on Combinations of Characters and Word Levels

4.1. Our Approach. We built a Chinese-Vietnamese translation model based on the combination of CL and WL translations. For the WL translation, we divided it into two phases as follows: (1) statistical translation of the WL and (2) the translation of SINO and NEs based on rule. In phase (1), some words are detected and translated into Chinese sentences. The results of this phase often are better than the results of these same words when they are translated in CL. One of hypotheses in phase (2) concerns the close relationship between the Chinese and Vietnamese languages. The statistical translation systems based on characters or words often mistranslated or did not translate NEs and SINO. Based on the relationship between the Chinese and Vietnamese languages [2], we developed some templates for the translation of NEs and SINO. The translation at the word WL often gives better results than the translation of the CL, but it generates many UKWs. Therefore, we continued to use the statistical translation system based on CL in order to translate words that could not be translated by the WL.

Our translation system is presented in Figure 2.

4.2. Training Process

4.2.1. Machine Translation Training Based on Statistics. The system uses SMT for both WL and CL. For WL, the Chinese corpus is segmented by the Stanford Segmenter toolkit, and the Vietnamese corpus is segmented by CLC_VN_WS of the Computational Linguistics Center (CLC). For CL, Chinese sentences have one space inserted between the characters. Similarly, one space is inserted between the spelling words and the punctuations in Vietnamese sentences. Both of the bilingual corpora are trained based on the following statistical formula [18]:

$$v' = \underset{v}{\operatorname{argmax}} p(v \mid c) \tag{1}$$

$$v' = \underset{v}{\operatorname{argmax}} p(v) * p(c \mid v), \tag{2}$$

where v is a Vietnamese sentence, c is a Chinese sentence, v' is the best Vietnamese sentence, $p(c \mid v)$ is the translation model, and $p(v)$ is the language model.

In this paper, we used phrase-based SMT, and formula (2) was modified as follows:

$$v' = \underset{v}{\operatorname{argmax}} \prod_{i=1}^{I} \phi(\overline{c_i} \mid \overline{v_i})\, d(\text{start}_i - \text{end}_{i-1} - 1) \cdot \prod_{i=1}^{|v|} P_{\text{LM}}(v_i \mid v_1 v_2 \cdots v_{i-1}), \tag{3}$$

where

$$\phi(\overline{c_i} \mid \overline{v_i}) = \frac{\text{count}(\overline{c_i}, \overline{v_i})}{\sum_{\overline{c_i}} \text{count}(\overline{c_i}, \overline{v_i})}, \tag{4}$$

where $\phi(\overline{c_i} \mid \overline{v_i})$ is phrase translation model, d is distortion model: $d(x) = \alpha^{|x|}$, $\alpha \in [0, 1]$, start_i is the position of the first word in phrase $\overline{e}$, end_i is the position of the last word in phrase $\overline{e}$, and $P_{\text{LM}}(v)$ is language model.

In practice, we used the Moses toolkit to train the corpora for both cases, including the word and character levels (phrase-based translation model and 3-gram language model).

4.2.2. Rule-Based Machine Translation Corpora. NEs and SINO are translated based on rule. The rule set of the system includes a CL Chinese-Sino-Vietnamese dictionary (cSINO dic), a word level Chinese-Sino-Vietnamese (wSINO), a list of Chinese family names (FN list), a keyword list of locations (LOC_KEY list), a list of the names of Chinese locations (LOC list), a list of the names of Chinese organizations (ORG list), and a set of rules for the translation of number expressions [19]. Figure 3 shows a sample of the dictionaries.

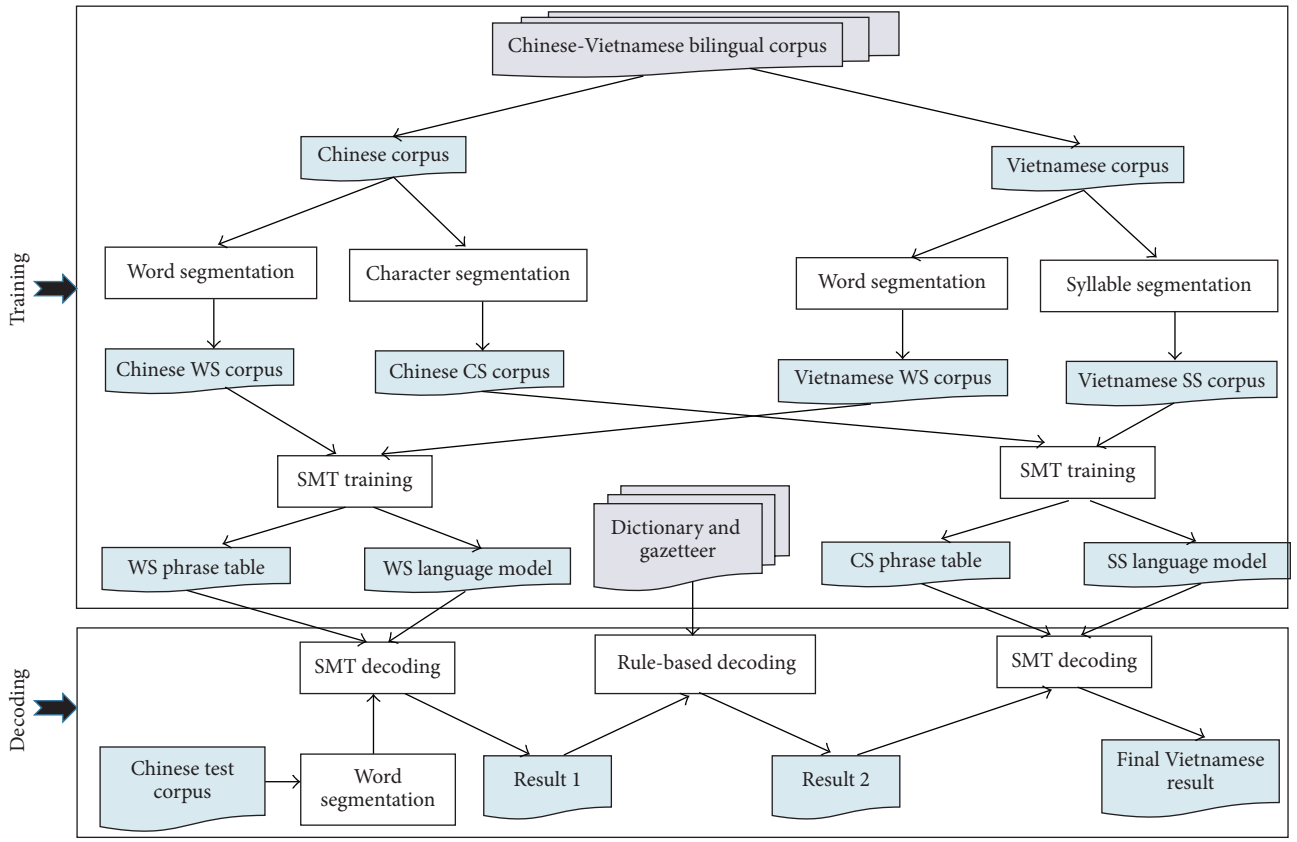


Figure 2: Our translation model.

(a) cSINO dic (b) wSINO dic (c) FN list

(d) LOC_KEY list (e) LOC list (f) ORG list

Figure 3: Samples of dictionaries in rule-based translation.

4.3. Decoding Phase. Our decoding phase is performed in three steps as follows.

4.3.1. Decoding Based on Statistics at WL. First, the Chinese corpus is segmented at WL and translated based on SMT decoding. In practice, we used Moses decoding to translate this corpus. At this stage, we do not identify NEs and SINO in the Chinese corpus. We consider the NEs and SINO as normal words, and the translation is based completely on statistics. This is done for the following two reasons:

(1) A Chinese word may have both a SINO meaning and a pure Vietnamese meaning, and the two meanings are synonyms of each other. For example, the Chinese word 现在 (now) is "hiện tại" in SINO, and its pure Vietnamese is "bây giờ," and the two meanings are equivalent. At this stage, the system translates the word 现在 completely based on statistics.

(2) The NEs in a Chinese document consist of Chinese NEs and foreign NEs. Typically, in the PER, if it is Chinese PER, the rule-based translation will give a correct result, but it will be incorrect if the PER is a foreign PER. For example, for the foreign PER, 奥巴马, the rule-based translation will give "Áo Ba Mã," while the correct translation is "Obama." Therefore, at this stage, we consider the NEs as normal words and translate them based on statistics.

4.3.2. Decoding Based on Rules. The words in Chinese sentences are translated based on rules, including NEs (PER, LOC, ORG, and NUM) and SINO. The three types, that is, PER, NUM, and SINO, appear frequently in Chinese corpus. In this phase, the system will perform two steps sequentially, that is, NE-SINO recognition and translation.

NE Recognition. There are many approaches for classifying NEs, and, in this work, we divided them into four categories, that is, PER, ORG, LOC, and NUM.

First, we used the Stanford Chinese NER toolkit to identify NEs in the Chinese corpus. Then, we used a heuristic algorithm and a set of rules to filter the NEs with four labels, that is, PER, LOC, ORG, and NUM. They were adjusted as follows:

(i) PER: Stanford NER toolkit annotates PERSON for both Chinese PER and foreign PER. In this work, based on the Chinese FN list, we keep only the PER tag for Chinese PER and untag foreign PER.

(ii) LOC: Stanford NER toolkit uses two tags, that is, LOC (Location) and GPE (geo-political entities), to annotate landmark, political entities. GPE indicates geographical or political names, such as cities, states, provinces, and countries. The two tags are quite close and have the same method to translate into Vietnamese, so we regroup them into a unique LOC tag.

(iii) ORG: The Stanford NER toolkit annotates the ORG tag for the organization names.

Based on the list of the names of Chinese locations and the list of the names of Chinese organizations, the system retains only the LOC and ORG tags for Chinese location or organization entities, and it untags the foreign LOC and ORG.

(i) NUM: Stanford NER toolkit annotates the MISC tag (Miscellaneous) to entities related to the number, time, or a combination of other NE and non-NE components. We change from MISC tag to NUM tag for temporal and numerical entities and remove the MISC tag for the remaining entities. Specifically, the system divides the NUM tag into seven labels [19], that is, numbers that contain unit characters, numbers that do not contain unit characters, ordinal numbers, fractions, decimal numbers, date, and time. What is more, the system retains only the NUMs that have two or more digits, because NUMs that have only one digit are limited, and the statistics-based translation system can deal with it successfully.

SINO Recognition. Based on word level Chinese-Vietnamese dictionary and the character-level Chinese-Sino-Vietnamese dictionary, the system assigns SINO tags for the Chinese corpus and Vietnamese corpus. We annotate only SINO tags for the words that have two or more syllables. However, because of some different SINO meanings in Chinese and Vietnamese (as mentioned in Section 3.1), we only assign a SINO tag for "pure Sino-Vietnamese words." The method is as follows.

Given that c is a Chinese word, V is a Vietnamese meaning set of c and S is a set of Sino-Vietnamese transliterations of the characters in c. Given $SN = V \cap S$, if $SN \neq \phi$, then c is labeled with a SINO tag.

For example, for the Chinese word 银行, we have V = {ngân hàng}, S = {ngân hành, ngân hạnh, ngân hàng, ngân hạng} and SN = {ngân hàng}; then the SINO tag will be assigned to 银行 (银行/SINO).

Rule-Based NE and SINO Translation. After identifying NE and SINO in a Chinese document, the system will translate these words based on rule. The translation method for each tag is presented below:

(1) PER: the Chinese family name will be recognized and translated based on the "FN list," and the Chinese given name is translated by "cSINO dic."

(2) LOC: LOC often takes the form ⟨location name⟩⟨keyword⟩. While ⟨Keyword⟩ is translated based on "LOC_KEY list", ⟨location name⟩ is translated based on "cSINO dic." The LOC and PER translation method is presented in detail in [20].

(3) ORG: this is the most complex type of NE, because it includes NE and non-NE components. Our system uses the method in [21] to translate ORG.

(4) NUM: the number expressions are translated based on the transformation rules between Chinese and Vietnamese. The rules are presented in [19].

Table 3: Distribution of number of words and number of sentences in experimental corpora of 11,000 sentence pairs.

Corpora		NS	CL		WL	
			NW	NW/NS	NS	NW/NS
Chinese	Training	9,900	99,026	10.0	72,541	7.3
	Developing	550	5,645	10.3	4,138	7.5
	Testing	550	5,598	10.2	4,092	7.4
Vietnamese	Training	9,900	107,153	10.8	93,909	9.5
	Developing	550	6,151	11.2	5,401	9.8
	Testing	550	5,985	10.9	5,272	9.6

Table 4: Distribution of number of words and number of sentences in experimental corpora of 22,000 sentence pairs.

Corpora		NS	CL		WL	
			NW	NW/NS	NW	NW/NS
Chinese	Training	19,800	196,903	9.9	144,475	7.3
	Developing	1,100	11,292	10.3	8,237	7.5
	Testing	1,100	11,056	10.1	8,090	7.4
Vietnamese	Training	19,800	211,179	10.7	185,346	9.4
	Developing	1,100	12,028	10.9	10,534	9.6
	Testing	1,100	11,803	10.7	10,376	9.4

Chinese sentence	你猜错 了，我 已经 七十六 了 。
Word level based SMT	Bạn 猜错 rồi, tôi đã 七十六 rồi.
Rule-based translation	Bạn 猜错 rồi, tôi đã **76** rồi.
Character level based SMT	Bạn **đoán sai** rồi, tôi đã **76** rồi.
Reference translation	Bạn **đoán sai** rồi, tôi đã **76** tuổi rồi.

Figure 4: Illustration about a translation of a Chinese sentence by our system.

(5) SINO: Sino-Vietnamese words in Chinese sentences are translated based on the word level Chinese-Sino-Vietnamese dictionary.

4.3.3. Decoding Based on Statistics at CL. In this step, the system translates the words that have yet to be translated for the first time by statistical decoding and rule-based decoding. These words include foreign NEs (PER, LOC, and ORG) and non-SINO words. They are decomposed into character CL and then translated based on statistics.

Figure 4 shows the translation of a Chinese sentence in the test corpus through the three phases of our system. (The English meaning of this sentence: "You guessed wrong; I am 76 years old.")

5. Experiments

5.1. Toolkit Used in Experiments. We used the Stanford Segmenter and the Stanford NER for WS and NE recognition in Chinese, and we used CLC_VN_NER and CLC_VN_WS for WS and NE recognition in Vietnamese.

In addition, we used the GIZA++ toolkit (download at http://www.fjoch.com/giza-training-of-statistical-translation-models.html) for word or character alignment, we used the SRILM toolkit (download at http://www.speech.sri.com/projects/srilm/download.html) to train the language model, and we used the Moses toolkit (download at http://www.statmt.org/moses/?n=Moses.Releases) to train the phrase-based SMT.

5.2. Experimental Corpora and Evaluation Methods. The experimental bilingual corpus includes 33,372 Chinese-Vietnamese sentence pairs, which were provided by the Computational Linguistics Center (CLC) (download sample corpus: http://www.clc.hcmus.edu.vn/?page_id=32). We used 90% of the sentences for training, 5% of the sentences for testing, and the remaining 5% of the sentences for developing. The corpus of 33,372 sentence pairs was divided into three parts, that is, Part 1, which included 11,000 sentence pairs; Part 2, which had 22,000 sentence pairs; and Part 3, which was comprised of 33,372 sentence pairs. The three corpora were used to perform four experiments, that is, CL translation, WL translation, Google Translate, and translation based on our system. Tables 3, 4, and 5 show the total number of sentences, its number of words, and number of words per sentence of the experimental corpus in which, for every 20 sentences, the 1st sentence to the 18th sentence is used for training set, the 19th sentence is used for developing, and the 20th sentence is used for testing.

5.3. Experimental Results. We divided several sentences in the experimental corpus into five cases. The BLEU metric is the average of these five cases. For every 20 sentences in the corpus, we distributed the sentences into the corpora as shown in Table 6.

Table 5: Distribution of number of words and number of sentences in experimental corpora of 33,372 sentence pairs.

Corpora		NS	CL		WL	
			NW	NW/NS	NW	NW/NS
Chinese	Training	30,036	301,630	10.0	221,419	7.4
	Developing	1,668	16,973	10.2	12,468	7.5
	Testing	1,668	17,049	10.2	12,453	7.5
Vietnamese	Training	30,036	316,453	10.5	278,232	9.3
	Developing	1,668	17,839	10.7	15,679	9.4
	Testing	1,668	17,745	10.6	15,617	9.4

NS is "number of sentences", NW is "number of words", and NW/NS is "NW per NS".
We used BLEU score and TER score to evaluate the performance of the translation systems.

Table 6: Distribution of number of sentences into the experimental corpora.

Case	Training corpus	Developing corpus	Testing corpus
1	From sentence 1 to sentence 18	Sentence 19	Sentence 20
2	From sentence 3 to sentence 20	Sentence 1	Sentence 2
3	From sentence 2 to sentence 19	Sentence 20	Sentence 1
4	From sentence 1 to sentence 10 and from sentence 31 to sentence 20	Sentence 11	Sentence 12
5	From sentence 1 to sentence 8 and from sentence 11 to sentence 20	Sentence 9	Sentence 10

Table 7: BLEU scores and TER scores of translation systems.

	CL		WL		Google translate		Our system	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
11,000	25.54	57.65	25.87	58.22	16.90	71.06	26.17	57.13
22,000	28.34	53.55	28.12	53.89	16.33	71.03	28.57	53.55
33,372	31.82	49.52	31.18	49.80	14.98	73.66	32.05	49.31

The BLEU scores and TER scores of each system are shown in Table 7. The MT output and the reference translation are segmented at the character/syllable level.

5.4. Analysis. The combination of statistics-based and rule-based translation at the CL and at the WL of our system aims to make use of the advantages of these methods. The advantages are include the local reordering of the phrase-based SMT, the precision of rule-based NE and SINO translation, and the capability of covering the characters of the CL translation system. The experimental results in Table 7 show that our system gives a better final translation than any of the other translation systems (as indicated by the BLEU scores and the TER scores).

For NE and SINO, the WL translation system will give the correct translation if they exist in the training corpus. Due to the limited training corpus, the WL translation system often gives many UKWs. If NE and SINO words are split into character level, the CL translation system can identify and translate them, but the results of the translation often are incorrect. For example, for the two words 王红 and 出色 in Figure 1, four characters, that is, 王, 红, 出, and 色, exist in the training corpus of the CL translation system. Their best meanings (or the highest translation probability) are "vương" (Vuong), "đỏ" (red), "ra" (to come out), and "màu" (color), respectively. In the four cases, only the 王/"Vương" pair is a correct translation. This error is due to the fact that the meaning at the CL and the meaning at the WL are not related to each other. So, the statistics-based translation at the CL for these cases is not feasible.

For these cases, the rule-based translation system will give better results. Based on the close relationship between the Chinese and Vietnamese languages, such as NE and SINO translation [2], we built the set of transformation rules in order to translate the NE and SINO, which could not translated by the WL translation system. For example, a Chinese PER 王红 must be translated into Sino-Vietnamese and must be uppercased. Its correct translation is "Vương Hồng" (not "vương đỏ" as in CL translation system). Similarly, the Sino-Vietnamese 出色 must be translated based on the Sino-Vietnamese transliteration. It means "xuất sắc" ("remarkable"), while the CL translation system gives the result as "ra màu."

However, the rule-based translation system only can deal with words that belong to SINO or NE categories. It cannot translate all of the UKW words that cannot be translated by the WL translation system (in Step 1). In this step, the CL translation system proved to be more effective. The words that the two previous systems were unable to translate will be decomposed into characters, and, then, they were translated by the CL translation system. In this phase, the translation of the Chinese document was relatively complete. Obviously, due to the lack of resources, UKWs still may appear in the phase, but number of UKWs will

not be more than that of both CL and WL translation systems.

As for Google Translate, this translation system has to translate twice when translating from Chinese into Vietnamese; the translation errors in Vietnamese side include the errors of the Chinese-English translation and English-Vietnamese translation. Moreover, Google Translate usually translates incorrectly NE words. A Chinese PER "王芳" is a good example, Google Translate transliterated it into "Wang Fang," (Pinyin transliteration) while the correct transliteration is "Vương Phương" (Sino-Vietnamese transliteration). So, in the three corpora (11,000; 22,000; and 33,372), Google Translate has the lowest BLEU scores and gives the highest TER scores.

6. Conclusions and Perspectives

In this paper, we built Chinese-Vietnamese translation models by combining the strengths of two approaches, that is, statistics-based and ruled-based translation on the two levels of characters and words. This approach is suitable to the language pairs that have a low-resource requirement, have a close relationship, and a space cannot determine the word boundary. As for the language pairs, although the WL translation system often translates word more accurately than the CL translation system, it generates more UKWs. Furthermore, regarding the languages in which characters are not the smallest unit of the vocabulary, the CL translation system will incorrectly translate words with meanings that are not related to the characters that form them. Typically, in Chinese-Vietnamese MT, the named entities and Sino-Vietnamese cannot be translated properly by the CL translation system.

The experimental results indicated that our combined system significantly improved the performance of MT over the performances of both the CL and WL translation systems. The improved performance of our system is reflected in different aspects, that is, it produces fewer UKWs than the WL translation system, translates NEs and SINO better than the CL translation system, and has a higher BLEU score than either the CL translation system or the WL translation system.

Given these results, we plan to integrate more linguistic information into the system in order to increase the quality of Chinese-Vietnamese MT.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

The authors sincerely thank the Computational Linguistics Center (CLC) that has contributed to them the Chinese-Vietnamese bilingual corpus.

References

[1] Y. M. Oh, F. Pellegrino, E. Marsico, and C. Coupé, "A quantitative and typological approach to correlating linguistic complexity," in *Proceedings of the 5th Conference on Quantitative Investigations in Theoretical Linguistics*, Leuven, Belgium, September 2013.

[2] L. Dinh Khan, *Vietnamese Vocabulary Having Chinese Origin*, National University of HCMC Press, 2002 (Vietnamese).

[3] Y.-S. Lee, "Morphological analysis for statistical machine translation," in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '04)*, pp. 57–62, 2004.

[4] S. Goldwater and D. McClosky, "Improving statistical MT through morphological analysis," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*, pp. 676–683, October 2005.

[5] J. Xu, R. Jens, and H. Ney, "Do we need Chinese word segmentation for statistical machine translation?" in *Proceedings of the 3rd SIGHAN Workshop on Chinese Language Learning*, pp. 122–128, Association for Computational Linguistics, Stroudsburg, Pa, USA, 2004.

[6] Y. Ma and A. Way, "Bilingually motivated domain-adapted word segmentation for statistical machine translation," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*, pp. 549–557, April 2009.

[7] M. Paul, A. Finch, and E. Sumita, "Integration of multiple bilingually-learned segmentation schemes into statistical machine translation," in *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR (WMT '10)*, pp. 400–408, Uppsala, Sweden, 2010.

[8] M.-H. Bai, K.-J. Chen, and J. S. Chang, "Improving word alignment by adjusting Chinese word segmentation," in *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, vol. 1, pp. 249–256, 2008.

[9] X. Wang, M. Utiyama, A. Finch, and E. Sumita, "Refining Word Segmentation Using a Manually Aligned Corpus for Statistical Machine Translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '14)*, pp. 1654–1664, Doha, Qatar, October 2014.

[10] C. Chu, T. Nakazawa, D. Kawahara, and S. Kurohashi, "Exploiting shared Chinese characters in Chinese word segmentation optimization for Chinese-Japanese machine translation," in *Proceeding of the 16th Annual Conference of the European Association for Machine Translation*, pp. 35–42, 2012.

[11] J. Xu, E. Matusov, R. Zens, and H. Ney, "Integrated Chinese word segmentation in statistical machine translation," in *Proceeding of the International Workshop on Spoken Language Translation*, pp. 131–137, 2005.

[12] H. Zhao, T. Yin, and J. Zhang, "Vietnamese to Chinese machine translation via Chinese character as pivot," in *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC '13)*, pp. 250–259, Taipei, Taiwan, November 2013.

[13] H. Zhao, M. Utiyama, E. Sumita, and B.-L. Lu, "An empirical study on word segmentation for Chinese machine translation," in *Computational Linguistics and Intelligent Text Processing*, vol. 7817 of *Lecture Notes in Computer Science*, pp. 248–263, Springer, Berlin, Germany, 2013.

[14] J. Xu, J. Gao, K. Toutanova, and H. Ney, "Bayesian semi-supervised Chinese word segmentation for statistical machine translation," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling '08)*, pp. 1017–1024, August 2008.

[15] T. L. Nguyen, S. Vogel, and N. A. Smith, "Nonparametric word segmentation for machine translation," in *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*, pp. 815–823, August 2010.

[16] X. Wang, M. Utiyama, A. Finch, and E. Sumita, "Empirical study of unsupervised Chinese word segmentation methods for SMT on large-scale corpora," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL '14)*, pp. 752–758, June 2014.

[17] X. Zeng, L. S. Chao, D. F. Wong, I. Trancoso, and L. Tian, "Toward better Chinese word segmentation for SMT via bilingual constraints," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 1360–1369, Association for Computational Linguistics, June 2014.

[18] P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: parameter estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

[19] P. Tran and D. Dinh, "Retranslating number expression unknown word in Chinese-Vietnamese statistical machine translation," *Journal of Computer Science and Cybernetics*, vol. 30, no. 2, pp. 127–138, 2014 (Vietnamese).

[20] P. Tran, D. Dinh, and L. Tran, "Resolving named entity unknown word in Chinese-Vietnamese machine translation," in *Proceedings of the Fifth International Conference on Knowledge and Systems Engineering (KSE '13)*, pp. 273–285, 2013.

[21] P. Tran, D. Dinh, T. Le, and T. Nguyen, "Handling organization name unknown word in Chinese-Vietnamese machine translation," in *Proceedings of the IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF '13)*, pp. 242–247, IEEE, Hanoi, Vietnam, November 2013.