International Journal of Software Engineering and Its Applications
Vol. 10, No. 2 (2016), pp. 93-104
http://dx.doi.org/10.14257/ijseia.2016.10.2.08

Named Entity Recognition using Word Embedding as a Feature[1]

Miran Seok[1,2], Hye-Jeong Song[1,2], Chan-Young Park[1,2],
Jong-Dae Kim[1,2], Yu-seop Kim[1,22]

[1]*Department of Convergence Software, Hallym University, Korea*
[2]*Bio-IT Research Center, Hallym University, Korea*
smr4880@hanmail.net, {hjsong, cypark, kimjd, yskim01}@hallym.ac.kr

Abstract

This study applied word embedding to feature for named entity recognition (NER) training, and used CRF as a learning algorithm. Named entities are phrases that contain the names of persons, organizations and locations and recognizing these entities in text is one of the important task of information extraction. Word embedding is helpful in many learning algorithms of NLP, indicating that words in a sentence are mapped by a real vector in a low-dimension space. We used GloVe, Word2Vec, and CCA as the embedding methods. The Reuters Corpus Volume 1 was used to create word embedding and the 2003 shared task corpus (English) of CoNLL was used for training and testing. As a result of comparing the performance of multiple techniques for word embedding to NER, it was found that CCA (85.96%) in Test A and Word2Vec (80.72%) in Test B exhibited the best performance. When using the word embedding as a feature of NER, it is possible to obtain better results than baseline that do not use word embedding. Also, to check that the word embedding well performed, we did additional experiment calculating the similarity between words.

Keywords*: Natural Language Processing, Named Entity Recognition, Word Embedding*

1. Introduction

The first research in the field of named entity recognition (NER) [1] is on a system that can be used to recognize and extract company names, depending on heuristics and handcrafted rules. But most recent studies use the rule-based system or supervised machine learning as a method to automatically execute a sequence labeling algorithm. Supervised learning techniques include Hidden Markov Models (HMMs) [2-4], Decision Trees [5], Maximum Entropy Models [6-10], Support Vector Machines (SVMs) [11-13], and Conditional Random Fields (CRFs) [14-18]. These supervised learning methods generally consist of systems that read a large annotated corpus, remember a list of entities, and create clear rules based on differentiated features.

This study used CRF as a learning algorithm. CRFs are a class of statistical modelling method and predicts sequences of labels for sequences of input samples. Many different classifiers predict the labels for a single argument without regard to the neighboring constituents, on the other hand, CRF model predicts its label considering the neighboring words.

Features are more important than model. NER systems often use different types of linguistic features including morphological, syntactic and semantic information of words. Recently, there has been great interest in using massive unlabeled data to learn word

[1] This paper is a revised and expanded version of a paper entitled [Comparison of NER Performance Using Word Embeddings] presented at [The 4th International Conference on Artificial Intelligence and Application, Jeju National University International Center, Jeju Island, Korea and December 16-19, 2015].
[2] He is a corresponding author

ISSN: 1738-9984 IJSEIA

representation that used as a feature in supervised classifiers for a natural language processing (NLP) task. Word embedding is helpful in many learning algorithms of NLP, such as machine translation and voice recognition [19]. This study added unsupervised word representation as a feature to improve the performance of the existing supervised NLP system. Word embedding is a feature learning technique in NLP, indicating that words in a sentence are mapped by a real vector in a low-dimension space. Because word embedding has semantic or grammatical interpretations, this study uses word embedding, including Global Vector (GloVe), Word2Vec and Canonical Correlation Analysis (CCA), as a feature for learning NER.

As a result, all the three word embedding algorithms improve the performance of NER task. When comparing the word embedding methods, it was found that CCA in Test A, that is 85.96%, and Word2Vec in Test B (80.72%) exhibited the best performance. They showed an improvement in performance of approximately 1.8% and 3.6%, respectively, compared to the baseline.

This paper is organized as follows. Section 2 describes the background knowledge of named entity recognition and data where we used for NER experiments. Section 3 describes the word embedding methods. Section 4 presents the baseline features for NER and added word embedding feature. Section 5 discuss a conditional random field and section6 describes the experiments that including evaluation method and results. Finally, section 7 discuss the conclusion.

2. Named Entity Recognition

2.1. Summary

The most pertinent information in a document is typically revealed in the names that occur within it. The term “named entity” was created by the Sixth Message Understanding Conference (MUC-6). Named entity (NE) refers to phrases that include the names of persons, organizations, and locations, and so on. Table 1 shows some of the commonly used types of NEs.

Table 1. Commonly Used Types of Named Entity

NE Type	Examples
PERSON	Eddy Bonte, President Obama
ORGANIZATION	Georgia-Pacific Corp., WHO
LOCATION	Murray River, Mount Everest
DATE	June, 2008-06-29
TIME	two fifty a.m., 1:30 p.m.
MONEY	175 million Canadian Dollars, GBP 10.40
PERCENT	twenty pct, 18.75 %

In general, three proper names of “persons,” “locations,” and “organizations” were often studied as entity types, and the “miscellaneous” type was used in Computational Natural Language Learning (CoNLL) conferences. The “miscellaneous” type refers to proper nouns that are not included in the three aforementioned types.

NER classifies all words of a document in predefined categories (such as the names of persons, organizations, locations, expressions of time, quantities, monetary values, and percentages). NER, generally covered as a sequence prediction issue, is a highly important phase in extracting and managing intelligence information and is efficient in establishing a relatively simple and rational system.

Most studies on NER are conducted by tagging unannotated blocks in the text. For Example:

[**Jim**]$_{\text{Person}}$ bought 300 shares of [**Acme Corp.**]$_{\text{Organization}}$ in [**2006**]$_{\text{Time}}$.

The name of the person consists of a single token and the name of the company consists of two tokens, and a time expression is detected.

2.2. Data

This study used the 2003 shared task corpus (English) of CoNLL for NER learning. The data is a collection of Newswire articles from the Reuters Corpus, and annotation was performed by members of the University of Antwerp.

The CoNLL-2003 shared task data was applied linguistic preprocessing for tokenization, part-of-speech tagging and chunking. The data file (see Figure 1) includes four columns separated by space. Each word is placed on separate lines, and there are empty lines between the sentences.

```
U.N.        NNP  I-NP  I-ORG
official    NN   I-NP  O
Ekeus       NNP  I-NP  I-PER
heads       VBZ  I-VP  O
for         IN   I-PP  O
Baghdad     NNP  I-NP  I-LOC
.           .    O     O
```

Figure 1. Example of CoNLL-2003 Shared Task Data File

The first item of each line is the word, the second item is the part of speech, the third item is the syntactic chunk tag, and the fourth item is the named entity tag.

The chunk tag and named entity tag take IOB representation (i.e. inside, outside and beginning). The form of I-TYPE in which the words are inside the type. Only when two phrases with the same type follow each other, the first word of the second phrase has the B-TYPE tag to show that a new phrase is beginning. The O tag refers to tokens that do not belong to a certain named entity.

3. Word Embedding

In general, supervised lexicalized NLP approaches take a word and convert it to a feature vector using a one-hot representation [20]. The feature vector has the same size as the vocabulary size, and is made up of one dimension. The element in the word index is 1, and the other component is 0.

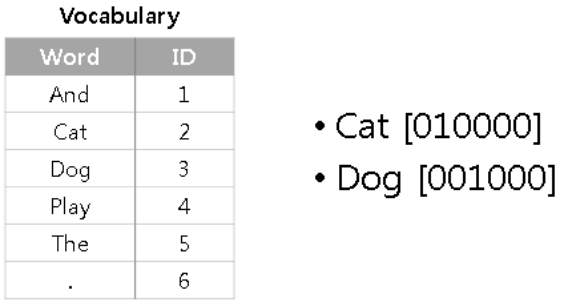

Figure 2. Example of One-Hot Representation

However, one-hot representation has scarcity of labeled data. In other words, there is a disadvantage of not being able to process words that do not appear in the labeled training data during the test.

Word embedding mapping words with vectors of real numbers, for example, W("cat") = (0.2, -0.4, 0.7, ⋯) where W is word embedding and W("cat") is word embedding of word "cat".

Word Embedding can help make better performance in natural language processing by grouping similar words. In word embedding, similar words are likely to have similar vectors.

Word embedding can capture various dimensions of meanings and phrase information relevant to the potential features of words within the vector. As a result, word embedding is less sensitive to data scarcity.

There are various methods of word embedding and this study established word embedding using GloVe, Word2Vec, and CCA.

3.1. Global Vector

The statistics of word occurrences in a corpus is the primary source of information available to all unsupervised methods for learning word representations [21]. GloVe is an unsupervised learning algorithm to obtain vector representations of words. Learning is carried out in global word-word co-occurrence statistics counted from the corpus.

The GloVe model learns items that are not 0 in the global word-word co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. This matrix shows how often the words co-occur in the given corpus in a table.

In general, the number of matrix entries that are not 0 is much smaller than the total number of words in the corpus; thus, the following training iterations are progressed much more quickly. A weighted least-squares regression model takes the form:

$$\sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \quad (1)$$

X is a word co-occurrence matrix, X_{ij} is the frequency of word i co-occurring with word j, and $X_i = \sum_k^V X_{ik}$ is the total number of occurrences of word i in the corpus. The probability of word j that occurs in the context of word i is $X_{ij} = P(j|i) = X_{ij}/X_i$. w is word embedding, and $\tilde{w}$ is separate context word embedding. A weight function $f(X_{ij})$ has the following desiderata: 1) $f(0) = 0$, 2) $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted, and 3) $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

For example, consider the co-occurrence probabilities for target words *ice* and *steam* with various probe words from the vocabulary. Here are some actual probabilities from a 6 billion word corpus:

Table 2. Co-Occurrence Probabilities and Ratios for a Large Corpus

Prob. and Ratio	k = solid	k = gas	k = water	k = fashion
P(k|ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k|steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k|ice) / P(k|steam)	8.9	8.5×10^{-2}	1.36	0.96

As one might expect, *ice* co-occurs more frequently with *solid* than it does with *gas*, whereas *steam* co-occurs more frequently with *gas* than it does with *solid*. Both words co-occur with their shared property *water* frequently, and both co-occur with the unrelated word *fashion* infrequently. In this way, the ratio of probabilities

encodes some crude form of meaning associated with the abstract concept of thermodynamic phase.

3.2. Word2Vec

Word2Vec is a linguistic model based on a neural network. Word2Vec learns the embedding of each word to map each discrete word into a low-dimensional continuous vector-space from their distributional properties observed in some raw text corpus, when a massive corpus is entered. The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector-space.

Word2Vec forms a simple log-linear classification network, and provides skip-gram architecture and continuous bag-of-words (CBOW) architecture.

The CBOW network predicts each word on the basis of neighboring words. The input layer of CBOW is represented as a bag (multiset) of words, order of words in the history does not influence the projection, and is projected in the hidden layer in a linear form. All words get projected into the same position, that is, vectors are averaged.

CBOW can be learned using extremely massive data that cannot be processed in another neural network bag-of-words model [22].

Skip-gram is a useful word representation in predicting neighboring words in a sentence or a document. It predicts the neighboring words or context when a single word is given. Skip-gram captures the averaged co-occurrence of two words in a training set. Unlike most of the previously used neural network architectures for learning word vectors, training of the skip-gram model does not involve dense matrix multiplications [23].

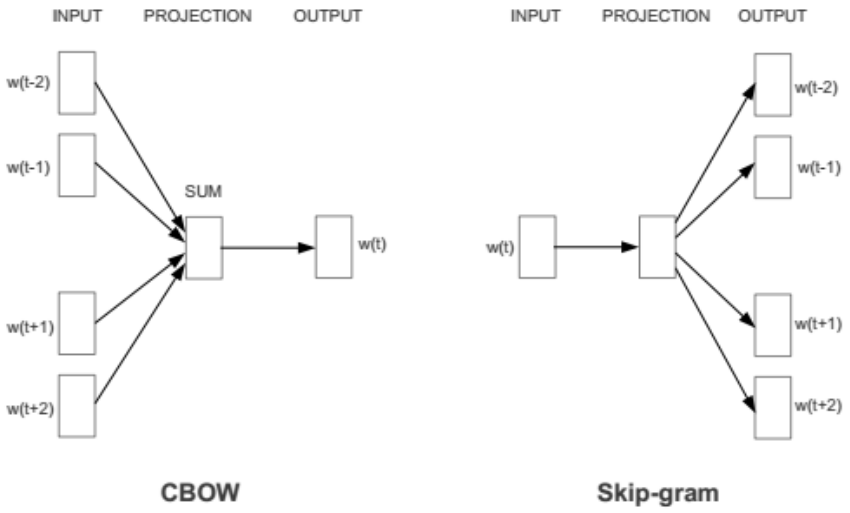

Figure 3. The Word2Vec Model Architecture [24]

Theoretically, CBOW must be superior because it includes more words that are right for the situation, but skip-gram actually has higher accuracy [1]. Therefore, this study established word embedding using the skip-gram model of Word2Vec.

When given the sequence of words $w_1, w_2, \dots, w_n$, the purpose of skip-gram is to maximize the average log probability

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{-c<j<c, j\neq 0}\log p(w_{n+j}|w_n) \quad (2)$$

where c is the size of the training context; the bigger the value of c, the better the learning results obtained. The inner summation goes from −c to c to compute the log probability of correctly predicting the word w_{n+j} given the word in the middle w_n. The outer summation goes over all words in the training corpus. [25]

The "input" and "output" vector representations of the w are u_w and v_w. The basic Skip-gram formulation defines $p(w_{n+j}|w_n)$ using the softmax function:

$$p(w_O|w_I) = \frac{\exp(u_{w_O}^{\top} v_{w_I})}{\sum_{k=1}^{V} \exp(u_k^{\top} v_{w_I})}$$

(3)

where V is size of the vocabulary [23].

3.3. Canonical Correlation Analysis (CCA)

Canonical correlation analysis is a statistical method used to investigate the relationship between two or more variable sets. CCA can be calculated directly from the data set, or calculated from representations such as covariance matrices. The algorithm for the two representations is based on singular value decomposition (SVD).

[26] and [27] give strong theoretical guarantees when the lower dimensional embeddings from CCA are used for predicting the label of the data point.

Suppose that n samples of the two variables are $x_1, \ldots, x_n \in \mathbb{R}^d$ and $y_1, \ldots, y_n \in \mathbb{R}^d$. For simplicity, suppose that these variables have zero mean. Then, CCA calculates the following equation for i $= 1, \ldots, d$.

$$\underset{u_i, y_i}{\operatorname{argmax}} \frac{\sum_{j=1}^{n} (u_i^{\top} x_j)(v_i^{\top} y_j)}{\sqrt{\sum_{j=1}^{n} (u_i^{\top} x_j)^2} \sqrt{\sum_{j=1}^{n} (v_i^{\top} y_j)^2}}$$

(4)

Under the precondition that the projection is uncorrelated with the previous $i-1$ projection, each (u_i, v_i) is a pair of projection vectors where correlation between projected variables $u_i^{\top} x_j$ and $v_i^{\top} y_j$ is maximum.

4. Feature Representation

This study used two feature sets of baseline and word embedding for NER.

4.1. Baseline Features

Table 3. Features Used in the Baseline System

Feature		Description
1	Unigram	$w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$
	Bigram	$w_{i-1}\|w_i$, $w_i\|w_{i-1}$
	POS	$POS_{i-2}, POS_{i-1}, POS_i, POS_{i+1}, POS_{i+2}$
	Named Entity	$NE_{i-1}, NE_{i-2}\|NE_{i-1}$
2	Prefix	Prefix of w_i, 0 < Prefix ≤ 4
	Suffix	Suffix of w_i, −4 ≤ Suffix < 0
	isDigit	Check if digit exists in w_i

	isUpper	Check if upper case exists in w_i
	isHyphen	Check if '-' exists in w_i

The first feature set in Table 3 consists of unigram and bigram of the words, unigram of parts of speech, and the unigram and bigram of the named entity tag using a window size of 2. These features are applied to all words.

The second feature set is applied to rare words that appear fewer than five times in the training data. The first two features encode the prefix and suffix that are less than four characters in length, and the remaining word features encode the presence or absence of each morphological characteristics. This feature set is known to be an important feature in tagging unknown words.

4.2. Word Embedding Features

This study used dimension values of word embedding as a new feature. The Reuters Corpus Volume 1, which consists of 800,000 English-language news stories that are manually classified, was used to create word embedding.

Embedding was established using GloVe, Word2Vec, and CCA, consisting of 44,532 words that appeared at least 101 times in the data. The dimension of each word vector is 50 and window size for context information was set as four words in front and four words in back, based on the current word.

5. Conditional Random Field

We used the Conditional random fields (CRF) model for the NER task because the model allow a great deal of flexibility in the features which can be included. CRFs represent a probabilistic framework for labeling and segmenting sequential data [16]. CRFs belong to a general class of algorithms known as undirected graphical models, and find favor over more classical Hidden Markov Models (HMM) as they are better able to account for dependency among specific events.

The input is a sentence consisting of a sequence of words $x_1, \cdots, x_n$; tags $y_1, \cdots, y_n$ suitable for each word (token) are allocated.

$$P(y|x) = \frac{1}{Z(x)} \prod_{i=1}^{n} exp \sum_{j} \lambda_j f_j(y_{i-1}, y_i, x, i) \tag{5}$$

In the equation, $f_j(y_{i-1}, y_i, x, i)$ is feature function that takes in as input: a sentence *x*, the position *i*, the label y_i of the current word, and the label y_{i-1} of the previous word. λ_j is weight learned about the feature, and *Z*(*x*) is a normalization function.

Higher λ weights make their corresponding finite state machine (FSM) transitions more likely, so the weight λ_j in this example should be positive.

The normalization factor, *Z*(*x*), is the sum of the "scores" of all possible state sequences:

$$Z(x) = \sum_y \exp\left(\sum_{i=1}^{n} \sum_j \lambda_j f_j(y_{i-1}, y_i, x, i)\right) \tag{6}$$

and that the number of state sequences is exponential in the input sequence length, N.

6. Experiments and Results

In this study, NER was learned using the average perceptron algorithm of CRFsuite, and the 2003 shared task corpus (English) of CoNLL was used for training and testing. Table 4 contains number of sentences and tokens in each data.

Table 4. Size of the data

	Sentences	Tokens
Training set	14,987	203,621
Test A	3,466	51,362
Test B	3,684	46,435

6.1. Evaluation

For the performance measurement of the NER task, the F1-score is more suitable than accuracy. Most labels in the NER data include the O tag, which refers to tokens and not a named entity, and thus high accuracy can be obtained. Therefore, this study measured performance using the harmonic mean of precision and recall.

Precision is the number of correct positive results divided by the number of all positive results, and recall is the number of correct positive results divided by the number of positive results that should have been returned. F1 score considers both the precision and the recall of the test to compute the score. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

6.2. NER Results

The experiment was performed in the case where the only baseline feature was used and the case where GloVe, Word2Vec, and CCA embeddings were added as features to the baseline. Table 5 shows the results of each case for NER task.

Table 5. Experimental F1-Score Results

	Test A	Test B
Baseline	84.12%	77.06%
Baseline + GloVe	85.18%	79.48%
Baseline + Word2Vec	85.89%	**80.72%**
Baseline + CCA	**85.96%**	80.68%

First, the baseline feature showed an F1-score of 84.12% in Test A, and 77.06% in Test B. When GloVe embedding is added as a feature, an approximately 1.06% and 2.4% performance increase was observed in Test A and B, respectively. When Word2Vec is used as a feature, both Test A and B obtained higher F1 scores than when using GloVe embedding. And when CCA embedding was used as a feature, the F1 score was the highest in Test A at 85.96%. Also, Test B exhibited the best performance in the case of Word2Vec.

6.3. Nearest Neighbors of Word Embedding

A simple way to investigate the learned representations is to find the closest words for a user-specified word. If a word vector truly represents lexically and

semantically potential features of words, their nearest neighbors must be associated. Table 6 shows the nearest neighbors of words included in the PER and ORG tags.

Table 6. Nearest Neighbors of each Word Sense

Tag	Word	Nearest Neighbors
PER	John	Michael, George, Richard, David, Jack, Paul, Stephen, Moore, Bruce, Peter, Jim
	German	French, Swedish, Belgian, Italian, Danish, Dutch, Austrian, Spanish, Finnish
ORG	Council	Committee, Convention, Rights, Organisation, Policy, Agreement, Conciliation
	Ltd.	Ltd, Holdings, Industries, Corporation, Pte, Limited, Co, Group, Company, Enterprises, affiliate

Nearest neighbors are calculated by comparing the cosine similarity between the embedding of each word (such as John, German, Council, and Ltd.) and the embedding of all other words in the vocabulary.

When the embedding value of word A and B is given, respectively, the degree of cosine similarity can be expressed as follows:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

(8)

Each nearest neighbor word has semantic consistency in Table 6. When a word similar to John in the PER tag is found, words relevant to the names of people like John (e.g. David, Tony, Richard) appeared as nearest neighbor words, and German could also obtain nearest neighbors of words in the same type.

Moreover, the nearest neighbors of “Council” of the ORG tag were found in similar words such as “Committee” and “Convention,” and it was found that “Organisation” was also included in the nearest neighbor. When words similar to “Ltd.” were searched, the same words “Ltd” and “Limited” were obtained, showing that multiple words related to company are included.

As such, word embedding well captures the semantic features of words, and well classifies and represents the words

7. Conclusion

When embedding was established using various word embedding methods and used as a feature for NER, all three algorithms showed improved results than using the baseline feature only. Especially, CCA and Word2Vec was obtained highest F1 score in each test. Moreover, the word expression ability of word embedding could be found through the nearest neighbors.

In the future, we can improve the F1-score by combining multiple techniques of word embedding features.

Once learned word embedding has the benefit of being able to be used in other research, and can be easily integrated with the supervised NLP system that already exists. In addition to NER, word embedding can be applied to multiple NLP tasks to obtain improved results.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning (2015R1A2A2A01007333).

References

[1] L.F. Rau, "Extracting Company Names from Text", Proc. Conference on Artificial Intelligence Applications of IEEE, **(1991)**.
[2] D.M. Bikel, S. Miller, R. Schwartz and R. Weischedel, "Recent Nymble: a High-Performance Learning Name-finder", Proceedings of the fifth conference on Applied natural language processing, **(1997)**, March.
[3] G. Zhou and J. Su, "Named entity recognition using an HMM-based chunk tagger", Proceedings of the 40th Annual Meeting Association for Computational Linguistics, **(2002)**.
[4] S. Zhao, "Named entity recognition in biomedical texts using an HMM model", Proceedings of the international joint workshop on natural language processing in biomedicine and its applications. Association for Computational Linguistics, **(2004)**.
[5] S. Sekine, "Nyu: Description of the Japanese NE System Used For Met-2", Proc. Message Understanding Conference, **(1998)**.
[6] A. Borthwick, J. Sterling, E. Agichtein and R. Grishman, "NYU: Description of the MENE Named Entity System as used in MUC-7", Proceedings of the Seventh Message Understanding Conference, **(1998)**.
[7] J.R. Curran and S. Clark, "Language independent NER using a maximum entropy tagger", Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. Association for Computational Linguistics, **(2003)**.
[8] O. Bender, F.J. Och and H. Ney, "Maximum entropy models for named entity recognition", Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. Association for Computational Linguistics, **(2003)**.
[9] Y. Benajiba, P. Rosso and J.M. Benedíruiz, "Anersys: An Arabic named entity recognition system based on maximum entropy", Computational Linguistics and Intelligent Text Processing. Springer Berlin Heidelberg, **(2007)**.
[10] I. Ahmed and R. Sathyaraj, "Named Entity Recognition by Using Maximum Entropy", International Journal of Database Theory & Application., vol. 8, no. 2, **(2015)**.
[11] J. Kazama, T. Makino, Y. Ohta and J. Tsujii, "Tuning Support Vector Machines for Biomedical Named Entity Recognition", Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain-Volume 3. Association for Computational Linguistics, **(2002)**.
[12] K.J. Lee, Y.S. Hwang and H.C. Rim, "Two Phase Biomedical NE Recognition based on SVMs", Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13. Association for Computational Linguistics, **(2003)**.
[13] T.D. Singh and S. Bandyopadhyay, "Web Based Manipuri Corpus for Multiword NER and Reduplicated MWEs Identification using SVM", Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing (WSSANLP), **(2010)**.
[14] A. McCallum and W. Li, "Early Results for Named Entity Recognition with Conditional Random Fields, Features Induction and Web-Enhanced Lexicons", Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. Association for Computational Linguistics, **(2003)**.
[15] Y. Song, E. Kim, G.G. Lee and B.K. Yi, "POSBIOTM-NER in the shared task of BioNLP/NLPBA 2004", Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications. Association for Computational Linguistics, **(2004)**, August.
[16] Z. Xu, X. Qian, Y. Zhang and Y. Zhou, "CRF-based Hybrid Model for Word Segmentation, NER and even POS Tagging", IJCNLP, **(2008)**, pp. 167-170.
[17] A. Ekbal and S. Bandyopadhyay, "Voted NER system using appropriate unlabeled data", Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration. Association for Computational Linguistics, **(2009)**, August.
[18] M. Konkol and M. Konopík, "Crf-based czech named entity recognizer and consolidation of czech ner research", Text, Speech, and Dialogue. Springer Berlin Heidelberg, **(2013)**, January.
[19] L. Qiu, Y. Cao, Z. Nie and Y. Rui, "Learning Word Representation Considering Proximity and Ambiguity", Twenty-Eighth AAAI Conference on Artificial Intelligence, **(2014)**.
[20] J. Turian, L. Ratinov and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning", Proceedings of the 48th annual meeting of the association for computational linguistics, **(2010)**, July.

[21] J. Pennington, R. Socher and C.D. Manning, "Glove: Global vectors for word representation", Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 12, **(2014)**, pp. 1532-1543.
[22] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A neural probabilistic language model", The Journal of Machine Learning Research., vol. 3, **(2003)**, pp. 1137-1155.
[23] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality", Advances in neural information processing systems, **(2013)**.
[24] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space", arXiv preprint arXiv:1301.3781, **(2013)**.
[25] T. Mikolov, Q.V. Le and I. Sutskever, "Exploiting similarities among languages for machine translation", **(2013)**.
[26] S.M. Kakade and D.P. Foster, "Multi-view regression via canonical correlation analysis", Learning Theory. Springer Berlin Heidelberg, **(2007)**, pp. 82-96.
[27] K. Sridharan and S.M. Kakade, "An Information Theoretic Framework for Multi-view Learning", Conference on Learning Theory (COLT), **(2008)**, pp. 403-414.
[28] M.R. Seok, H.J. Song, C.Y. Park, J.D. Kim and Y.S. Kim, "Comparison of NER Performance Using Word Embeddings", The 4th International Conference on Artificial Intelligence and Application, **(2015)**, December 16-19.

Authors

Miran Seok, She received the B.S. degree in Department of Ubiquitous Computing from Hallym University. He currently studies for a master's degree in Department of Convergence Software at Hallym University. Her recent interests focus on natural language processing.

Hye-Jeong Song, She received the Ph.D. degree in Computer Engineering from Hallym University. She is a Professor in Department of Convergence Software, Hallym University. Her recent interests focus on biomedical system and bioinformatics.

Chan-Young Park, He received the B.S. and the M.S. from Seoul National University and the Ph.D. degree from Korea Advanced Institute of Science and Technology in 1995. From 1991 to 1999, he worked at Samsung Electronics. He is currently a Professor in the Department of Convergence Software of Hallym University, Korea. His research interests are in Bio-IT convergence, Intelligent Transportation System and sensor networks.

Jong-Dae Kim, He received the M.S. and the Ph.D. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology, Seoul, Korea, in 1984 and 1990, respectively. He worked for Samsung Electronics from 1988 to 2000 as an electrical engineer. He is a Professor in Department of Convergence Software, Hallym University. His recent interests focus on biomedical system and bioinformatics.

Yu-seop Kim, He received the Ph.D. degree in Computer Engineering from Seoul National University. He is currently a Professor in the Department of Convergence Software at Hallym University, South Korea. His research interests are in the areas of bioinformatics, computational intelligence and natural language processing.