Learning Condensed Feature Representations from Large Unsupervised Data Sets for Supervised Learning

Jun Suzuki, Hideki Isozaki, and Masaaki Nagata
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{suzuki.jun, isozaki.hideki, nagata.masaaki}@lab.ntt.co.jp

Abstract

This paper proposes a novel approach for effectively utilizing unsupervised data in addition to supervised data for supervised learning. We use unsupervised data to generate informative 'condensed feature representations' from the original feature set used in supervised NLP systems. The main contribution of our method is that it can offer dense and low-dimensional feature spaces for NLP tasks while maintaining the state-of-the-art performance provided by the recently developed high-performance semi-supervised learning technique. Our method matches the results of current state-of-the-art systems with very few features, *i.e.*, F-score 90.72 with 344 features for CoNLL-2003 NER data, and UAS 93.55 with 12.5K features for dependency parsing data derived from PTB-III.

1 Introduction

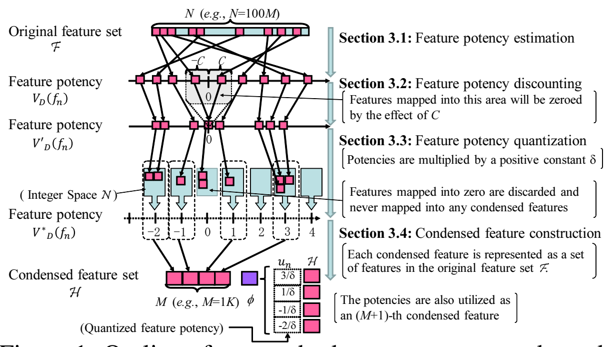
In the last decade, supervised learning has become a standard way to train the models of many natural language processing (NLP) systems. One simple but powerful approach for further enhancing the performance is to utilize a large amount of unsupervised data to supplement supervised data. Specifically, an approach that involves incorporating 'clustering-based word representations (**CWR**)' induced from unsupervised data as additional features of supervised learning has demonstrated substantial performance gains over state-of-the-art supervised learning systems in typical NLP tasks, such as named entity recognition (Lin and Wu, 2009; Turian et al., 2010) and dependency parsing (Koo et al., 2008). We refer to this approach as the **iCWR approach**, The iCWR approach has become popular for enhancement because of its simplicity and generality.

The goal of this paper is to provide yet another simple and general framework, like the iCWR approach, to enhance existing state-of-the-art supervised NLP systems. The differences between the iCWR approach and our method are as follows; suppose $\mathcal{F}$ is the original feature set used in supervised learning, $\mathcal{C}$ is the CWR feature set, and $\mathcal{H}$ is the new feature set generated by our method. Then, with the iCWR approach, $\mathcal{C}$ is induced independently from $\mathcal{F}$, and used in addition to $\mathcal{F}$ in supervised learning, *i.e.*, $\mathcal{F} \cup \mathcal{C}$. In contrast, in our method $\mathcal{H}$ is directly induced from $\mathcal{F}$ with the help of an existing model already trained by supervised learning with $\mathcal{F}$, and used in place of $\mathcal{F}$ in supervised learning.

The largest contribution of our method is that it offers an architecture that can drastically reduce the number of features, *i.e.*, from 10M features in $\mathcal{F}$ to less than 1K features in $\mathcal{H}$ by constructing 'condensed feature representations (**COFER**)', which is a new and very unique property that cannot be matched by previous semi-supervised learning methods including the iCWR approach. One noteworthy feature of our method is that there is no need to handle sparse and high-dimensional feature spaces often used in many supervised NLP systems, which is one of the main causes of the data sparseness problem often encountered when we learn the model with a supervised leaning algorithm. As a result, NLP systems that are both compact and high-performance can be built by retraining the model with the obtained condensed feature set $\mathcal{H}$.

2 Condensed Feature Representations

Let us first define the **condensed feature set** $\mathcal{H}$. In this paper, we call the feature set generally used in supervised learning, $\mathcal{F}$, the **original feature set**. Let N and M represent the numbers of features in $\mathcal{F}$ and $\mathcal{H}$, respectively. We assume $M \leq N$, and generally $M \ll N$. A condensed feature $h_m \in \mathcal{H}$ is charac-


N (e.g., N=100M)
Original feature set $\mathcal{F}$
Section 3.1: Feature potency estimation
−C C
Feature potency $V_D(f_n)$
Section 3.2: Feature potency discounting
Features mapped into this area will be zeroed by the effect of C
0
Feature potency $V'_D(f_n)$
0
Section 3.3: Feature potency quantization
Potencies are multiplied by a positive constant δ
(Integer Space $\mathcal{N}$)
Features mapped into zero are discarded and never mapped into any condensed features
Feature potency $V^*_D(f_n)$
−2 −1 0 1 2 3 4
Section 3.4: Condensed feature construction
Each condensed feature is represented as a set of features in the original feature set $\mathcal{F}$.
u_n $\mathcal{H}$
Condensed feature set $\mathcal{H}$
3/δ 1/δ -1/δ -2/δ
M (e.g., M=1K) ϕ
The potencies are also utilized as an (M+1)-th condensed feature
(Quantized feature potency)
Figure 1: Outline of our method to construct a condensed feature set.

$$\bar{r}(\mathbf{x}) = \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} r(\mathbf{x},\mathbf{y})/|\mathcal{Y}(\mathbf{x})|.$$
$$V^+_{\mathcal{D}}(f_n) = \sum_{\mathbf{x}\in\mathcal{D}} f_n(\mathbf{x},\hat{\mathbf{y}})(r(\mathbf{x},\hat{\mathbf{y}}) - \bar{r}(\mathbf{x}))$$
$$V^-_{\mathcal{D}}(f_n) = -\sum_{\mathbf{x}\in\mathcal{D}} \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})\setminus\hat{\mathbf{y}}} f_n(\mathbf{x},\mathbf{y})(r(\mathbf{x},\mathbf{y}) - \bar{r}(\mathbf{x}))$$
$$R_n = \sum_{\mathbf{x}\in\mathcal{D}} \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} r(\mathbf{x},\mathbf{y}) f_n(\mathbf{x},\mathbf{y}),\quad A_n = \sum_{\mathbf{x}\in\mathcal{D}} \bar{r}(\mathbf{x}) \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f_n(\mathbf{x},\mathbf{y})$$

Figure 2: Notations used in this paper.

terized as a set of features in $\mathcal{F}$, that is, $h_m = S_m$ where $S_m \subseteq \mathcal{F}$. We assume that each original feature $f_n \in \mathcal{F}$ maps, at most, to one condensed feature h_m. This assumption prevents two condensed features from containing the same original feature, and some original features from not being mapped to any condensed feature. Namely, $S_m \cap S_{m'} = \emptyset$ for all m and m', where $m \neq m'$, and $\bigcup_{m=1}^{M} S_m \subseteq \mathcal{F}$ hold.

The value of each condensed feature is calculated by summing the values of the original features assigned to it. Formally, let $\mathcal{X}$ and $\mathcal{Y}$ represent the sets of all possible inputs and outputs of a target task, respectively. Let $\mathbf{x} \in \mathcal{X}$ be an input, and $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$ be an output, where $\mathcal{Y}(\mathbf{x}) \subseteq \mathcal{Y}$ represents the set of possible outputs given $\mathbf{x}$. We write the n-th feature function of the original features, whose value is determined by $\mathbf{x}$ and $\mathbf{y}$, as $f_n(\mathbf{x},\mathbf{y})$, where $n \in \{1,\dots,N\}$. Similarly, we write the m-th feature function of the condensed features as $h_m(\mathbf{x},\mathbf{y})$, where $m \in \{1,\dots,M\}$. We state that the value of $h_m(\mathbf{x},\mathbf{y})$ is calculated as follows: $h_m(\mathbf{x},\mathbf{y}) = \sum_{f_n \in S_m} f_n(\mathbf{x},\mathbf{y})$.

3 Learning COFERs

The remaining part of our method consists of the way to map the original features into the condensed features. For this purpose, we define the feature potency, which is evaluated by employing an existing supervised model with unsupervised data sets. Figure 1 shows a brief sketch of the process to construct the condensed features described in this section.

3.1 Self-taught-style feature potency estimation

We assume that we have a model trained by supervised learning, which we call the '**base supervised model**', and the original feature set $\mathcal{F}$ that is used in the base supervised model. We consider a case where the base supervised model is a (log-)linear model, and use the following equation to select the best output $\hat{\mathbf{y}}$ given $\mathbf{x}$:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \textstyle\sum_{n=1}^{N} w_n f_n(\mathbf{x},\mathbf{y}), \qquad (1)$$

where w_n is a model parameter (or weight) of f_n. Linear models are currently the most widely-used models and are employed in many NLP systems.

To simplify the explanation, we define function $r(\mathbf{x},\mathbf{y})$, where $r(\mathbf{x},\mathbf{y})$ returns 1 if $\mathbf{y} = \hat{\mathbf{y}}$ is obtained from the base supervised model given $\mathbf{x}$, and 0 otherwise. Let $\bar{r}(\mathbf{x})$ represent the average of $r(\mathbf{x},\mathbf{y})$ in $\mathbf{x}$ (see Figure 2 for details). We also define $V^+_{\mathcal{D}}(f_n)$ and $V^-_{\mathcal{D}}(f_n)$ as shown in Figure 2 where $\mathcal{D}$ represents the unsupervised data set. $V^+_{\mathcal{D}}(f_n)$ measures the positive correlation with the best output $\hat{\mathbf{y}}$ given by the base supervised model since this is the summation of all the (weighted) feature values used in the estimation of the one best output $\hat{\mathbf{y}}$ over all $\mathbf{x}$ in the unsupervised data $\mathcal{D}$. Similarly, $V^-_{\mathcal{D}}(f_n)$ measures the negative correlation with $\hat{\mathbf{y}}$. Next, we define $V_{\mathcal{D}}(f_n)$ as the feature potency of f_n: $V_{\mathcal{D}}(f_n) = V^+_{\mathcal{D}}(f_n) - V^-_{\mathcal{D}}(f_n)$.

An intuitive explanation of $V_{\mathcal{D}}(f_n)$ is as follows; if $|V_{\mathcal{D}}(f_n)|$ is large, the distribution of f_n has either a large positive or negative correlation with the best output $\hat{\mathbf{y}}$ given by the base supervised model. This implies that f_n is an informative and potent feature in the model. Then, the distribution of f_n has very small (or no) correlation to determine $\hat{\mathbf{y}}$ if $|V_{\mathcal{D}}(f_n)|$ is zero or near zero. In this case, f_n can be evaluated as an uninformative feature in the model. From this perspective, we treat $V_{\mathcal{D}}(f_n)$ as a measure of feature potency in terms of the base supervised model.

The essence of this idea, evaluating features against each other on a certain model, is widely used in the context of semi-supervised learning, *i.e.*, (Ando and Zhang, 2005; Suzuki and Isozaki, 2008; Druck and McCallum, 2010). Our method is rough and a much simpler framework for implementing this fundamental idea of semi-supervised learning developed for NLP tasks. We create a simple framework to achieve improved flexibility, extendability, and applicability. In fact, we apply the framework by incorporating a feature merging and elimination architecture to obtain effective condensed feature sets for supervised learning.

3.2 Feature potency discounting

To discount low potency values, we redefine feature potency as $V'_{\mathcal{D}}(f_n)$ instead of $V_{\mathcal{D}}(f_n)$ as follows:

$$V'_{\mathcal{D}}(f_n) = \begin{cases} \log[R_n + C] - \log[A_n] & \text{if } R_n - A_n < -C \\ 0 & \text{if } -C \le R_n - A_n \le C \\ \log[R_n - C] - \log[A_n] & \text{if } C < R_n - A_n \end{cases}$$

where R_n and A_n are defined in Figure 2. Note that $V_{\mathcal{D}}(f_n) = V^{+}_{\mathcal{D}}(f_n) - V^{-}_{\mathcal{D}}(f_n) = R_n - A_n$. The difference from $V_{\mathcal{D}}(f_n)$ is that we cast it in the log-domain and introduce a non-negative constant C. The introduction of C is inspired by the L_1-regularization technique used in supervised learning algorithms such as (Duchi and Singer, 2009; Tsuruoka et al., 2009). C controls how much we discount $V_{\mathcal{D}}(f_n)$ toward zero, and is given by the user.

3.3 Feature potency quantization

We define $V^{*}_{\mathcal{D}}(f_n)$ as $V^{*}_{\mathcal{D}}(f_n) = \lceil \delta V'_{\mathcal{D}}(f_n) \rceil$ if $V'_{\mathcal{D}}(f_n) > 0$ and $V^{*}_{\mathcal{D}}(f_n) = \lfloor \delta V'_{\mathcal{D}}(f_n) \rfloor$ otherwise, where δ is a positive user-specified constant. Note that $V^{*}_{\mathcal{D}}(f_n)$ always becomes an integer, that is, $V^{*}_{\mathcal{D}}(f_n) \in \mathcal{N}$ where $\mathcal{N} = \{\dots, -2, -1, 0, 1, 2, \dots\}$. This calculation can be seen as mapping each feature into a discrete (integer) space with respect to $V'_{\mathcal{D}}(f_n)$. δ controls the range of $V'_{\mathcal{D}}(f_n)$ mapping into the same integer.

3.4 Condensed feature construction

Suppose we have M different quantized feature potency values in $V^{*}_{\mathcal{D}}(f_n)$ for all n, which we rewrite as $\{u_m\}_{m=1}^{M}$. Then, we define S_m as a set of f_n whose quantized feature potency value is u_m. As described in Section 2, we define the m-th condensed feature $h_m(\mathbf{x}, \mathbf{y})$ as the summation of all the original features f_n assigned to S_m. That is, $h_m(\mathbf{x}, \mathbf{y}) = \sum_{f_n \in S_m} f_n(\mathbf{x}, \mathbf{y})$. This feature fusion process is intuitive since it is acceptable if features with the same (similar) feature potency are given the same weight by supervised learning since they have the same potency with regard to determining $\hat{\mathbf{y}}$. δ determines the number of condensed features to be made; the number of condensed features becomes large if δ is large. Obviously, the upper bound of the number of condensed features is the number of original features.

To exclude possibly unnecessary original features from the condensed features, we discard feature f_n for all n if $u_n = 0$. This is reasonable since, as described in Section 3.1, a feature has small (or no) effect in achieving the best output decision in the base supervised model if its potency is near 0. C introduced in Section 3.2 mainly influences how many original features are discarded.

Additionally, we also utilize the 'quantized' feature potency values themselves as a new feature. The reason behind is that they are also very informative for supervised learning. Their use is important to further boost the performance gain offered by our method. For this purpose, we define $\phi(\mathbf{x}, \mathbf{y})$ as $\phi(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^{M} (u_m/\delta) h_m(\mathbf{x}, \mathbf{y})$. We then use $\phi(\mathbf{x}, \mathbf{y})$ as the $(M+1)$-th feature of our condensed feature set. As a result, the condensed feature set obtained with our method is represented as $\mathcal{H} = \{h_1(\mathbf{x}, \mathbf{y}), \dots, h_M(\mathbf{x}, \mathbf{y}), \phi(\mathbf{x}, \mathbf{y})\}$.

Note that the calculation cost of $\phi(\mathbf{x}, \mathbf{y})$ is negligible. We can calculate the linear discriminant function $g(\mathbf{x}, \mathbf{y})$ as: $g(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^{M} w_m h_m(\mathbf{x}, \mathbf{y}) + w_{M+1}\phi(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^{M} w'_m h_m(\mathbf{x}, \mathbf{y})$, where $w'_m = (w_m + w_{M+1} u_m/\delta)$. We emphasize that once $\{w_m\}_{m=1}^{M+1}$ are determined by supervised learning, we can calculate w'_m in a preliminary step before the test phase. Thus, our method also takes the form of a linear model. The number of features for our method is essentially M even if we add ϕ.

3.5 Application to Structured Prediction Tasks

We modify our method to better suit structured prediction problems in terms of calculation cost. For a structured prediction problem, it is usual to decompose or factorize output structure $\mathbf{y}$ into a set of local sub-structures z to reduce the calculation cost and to cope with the sparsity of the output space $\mathcal{Y}$. This factorization can be accomplished by restricting features that are extracted only from the information within decomposed local sub-structure z

and given input $\mathbf{x}$. We write $z \in \mathbf{y}$ when the local sub-structure z is a part of output $\mathbf{y}$, assuming that output $\mathbf{y}$ is constructed by a set of local sub-structures. Then formally, the n-th feature is written as $f_n(\mathbf{x}, z)$, and $f_n(\mathbf{x}, \mathbf{y}) = \sum_{z \in \mathbf{y}} f_n(\mathbf{x}, z)$ holds. Similarly, we introduce $r(\mathbf{x}, z)$, where $r(\mathbf{x}, z) = 1$ if $z \in \hat{\mathbf{y}}$, and $r(\mathbf{x}, z) = 0$ otherwise, namely $z \notin \hat{\mathbf{y}}$.

We define $\mathcal{Z}(\mathbf{x})$ as the set of all local sub-structures possibly generated for all $\mathbf{y}$ in $\mathcal{Y}(\mathbf{x})$. $\mathcal{Z}(\mathbf{x})$ can be enumerated easily, unless we use typical first- or second-order factorization models by the restriction of efficient decoding algorithms, which is the typical case for many NLP tasks such as named entity recognition and dependency parsing.

Finally, we replace all $\mathcal{Y}(\mathbf{x})$ with $\mathcal{Z}(\mathbf{x})$, and use $f_n(\mathbf{x}, z)$ and $r(\mathbf{x}, z)$ instead of $f_n(\mathbf{x}, \mathbf{y})$ and $r(\mathbf{x}, \mathbf{y})$, respectively, in R_n and A_n. When we use these substitutions, there is no need to incorporate an efficient algorithm such as dynamic programming into our method. This means that our feature potency estimation can be applied to the structured prediction problem at low cost.

3.6 Efficient feature potency computation

Our feature potency estimation described in Section 3.1 to 3.3 is highly suitable for implementation in the MapReduce framework (Dean and Ghemawat, 2008), which is a modern distributed parallel computing framework. This is because R_n and A_n can be calculated by the summation of a data-wise calculation (map phase), and $V^*_{\mathcal{D}}(f_n)$ can be calculated independently by each feature (reduce phase). We emphasize that our feature potency estimation can be performed in a 'single' map-reduce process.

4 Experiments

We conducted experiments on two different NLP tasks, namely NER and dependency parsing. To facilitate comparisons with the performance of previous methods, we adopted the experimental settings used to examine high-performance semi-supervised NLP systems; *i.e.*, NER (Ando and Zhang, 2005; Suzuki and Isozaki, 2008) and dependency parsing (Koo et al., 2008; Chen et al., 2009; Suzuki et al., 2009). For the supervised datasets, we used CoNLL'03 (Tjong Kim Sang and De Meulder, 2003) shared task data for NER, and the Penn Treebank III (PTB) corpus (Marcus et al., 1994) for dependency parsing. We prepared a total of 3.72 billion token text data as unsupervised data following the instructions given in (Suzuki et al., 2009).

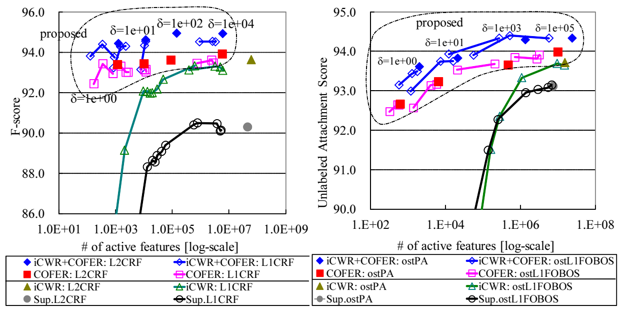
4.1 Comparative Methods

We mainly compare the effectiveness of COFER with that of CWR derived by the Brown algorithm. The iCWR approach yields the state-of-the-art results with both dependency parsing data derived from PTB-III (Koo et al., 2008), and the CoNLL'03 shared task data (Turian et al., 2010). By comparing COFER with iCWR we can clarify its effectiveness in terms of providing better features for supervised learning. We use the term **active features** to refer to features whose corresponding model parameter is non-zero after supervised learning. It is well-known that we can discard non-active features from the trained model without any loss after finishing supervised learning. Finally, we compared the performance in terms of the number of active features in the model given by supervised learning. We note here that the number of active features for COFER is the number of features h_m if $w'_m = 0$, which is not $w_m = 0$ for a fair comparison.

Unlike COFER, iCWR does not have any architecture to winnow the original feature set used in supervised learning. For a fair comparison, we prepared L_1-regularized supervised learning algorithms, which try to reduce the non-zero parameters in a model. Specifically, we utilized L_1-regularized CRF (**L1CRF**) optimized by OWL-QN (Andrew and Gao, 2007) for NER, and the online structured output learning version of FOBOS (Duchi and Singer, 2009; Tsuruoka et al., 2009) with L_1-regularization (**ostL1FOBOS**) for dependency parsing. In addition, we also examined L_2 regularized CRF (Lafferty et al., 2001) optimized by L-BFGS (Liu and Nocedal, 1989) (**L2CRF**) for NER, and the online structured output learning version of the Passive-Aggressive algorithm (**ostPA**) (Crammer et al., 2006) for dependency parsing to illustrate the baseline performance regardless of the active feature number.

4.2 Settings for COFER

We utilized baseline supervised learning models as the base supervised models of COFER.



(a) NER (F-score) (b) dep. parsing (UAS)

Figure 3: Performance vs. size of active features in the trained model on the development sets

In addition, we also report the results when we treat iCWR as COFER's base supervised models (**iCWR+COFER**). This is a very natural and straightforward approach to combining these two.

We generally handle several different types of features such as words, part-of-speech tags, word surface forms, and their combinations. Suppose we have K different *feature types*, which are often defined by *feature templates*, *i.e.*, (Suzuki and Isozaki, 2008; Lin and Wu, 2009). In our experiments, we restrict the merging of features during the condensed feature construction process if and only if the features are the same feature type. As a result, COFER essentially consists of K different condensed feature sets. The numbers of feature types K were 79 and 30 for our NER and dependency parsing experiments, respectively. We note that this kind of feature partition by their types is widely used in the context of semi-supervised learning (Ando and Zhang, 2005; Suzuki and Isozaki, 2008).

4.3 Results and Discussion

Figure 3 displays the performance on the development set with respect to the number of active features in the trained models given by each supervised learning algorithm. In both NER and dependency parsing experiments, COFER significantly outperformed iCWR. Moreover, COFER was surprisingly robust in relation to the number of active features in the model. These results reveal that COFER provides effective feature sets for certain NLP tasks.

We summarize the noteworthy results in Figure 3, and also the performance of recent top-line systems for NER and dependency parsing in Table 1. Overall, COFER matches the results of top-line semi-supervised learning systems even though it uses far fewer active features.

NER system	dev.	test	#.USD	#.AF
Sup.L1CRF	90.40	85.08	0	0.57M
iCWR: L1CRF	93.33	89.99	3,720M	0.62M
COFER: L1CRF ($\delta = 1e+00$)	93.42	88.81	3,720M	359
($\delta = 1e+04$)	93.60	89.22	3,720M	2.46M
iCWR+COFER: ($\delta = 1e+00$)	94.39	90.72	3,720M	344
L1CRF ($\delta = 1e+04$)	**94.91**	**91.02**	3,720M	5.94M
(Ando and Zhang, 2005)	93.15	89.31	27M	N/A
(Suzuki and Isozaki, 2008)	**94.48**	89.92	1,000M	N/A
(Ratinov and Roth, 2009)	93.50	90.57	N/A	N/A
(Turian et al., 2010)	93.95	90.36	37M	N/A
(Lin and Wu, 2009)	N/A	**90.90**	700,000M	N/A

Dependency parser	dev.	test	#.USD	#.AF
ostL1FOBOS	93.15	92.82	0	6.80M
iCWR: ostL1FOBOS	93.69	93.49	3,720M	9.67M
COFER:ostL1FOBOS ($\delta = 1e+03$)	93.53	93.23	3,720M	20.7K
($\delta = 1e+05$)	93.91	93.71	3,720M	3.23M
iCWR+COFER: ($\delta = 1e+03$)	93.93	93.55	3,720M	12.5K
ostL1FOBOS ($\delta = 1e+05$)	**94.33**	**94.22**	3,720M	5.77M
(Koo and Collins, 2010)	93.49	93.04	0	N/A
(Martins et al., 2010)	N/A	93.26	0	55.25M
(Koo et al., 2008)	93.30	93.16	43M	N/A
(Chen et al., 2009)	N/A	93.16	43M	N/A
(Suzuki et al., 2009)	**94.13**	**93.79**	3,720M	N/A

Table 1: Comparison with previous top-line systems on test data. (#.USD: unsupervised data size. #.AF: the size of active features in the trained model.)

In addition, the combination of iCWR+COFER significantly outperformed the current best results by achieving a 0.12 point gain from 90.90 to 91.02 for NER, and a 0.43 point gain from 93.79 to 94.22 for dependency parsing, with only 5.94M and 5.77M features, respectively.

5 Conclusion

This paper introduced the idea of condensed feature representations (COFER) as a simple and general framework that can enhance the performance of existing supervised NLP systems. We also proposed a method that efficiently constructs condensed feature sets through discrete feature potency estimation over unsupervised data. We demonstrated that COFER based on our feature potency estimation can offer informative dense and low-dimensional feature spaces for supervised learning, which is theoretically preferable to the sparse and high-dimensional feature spaces often used in many NLP tasks. Existing NLP systems can be made more compact with higher performance by retraining their models with our condensed features.

References

Rie Kubota Ando and Tong Zhang. 2005. A High-Performance Semi-Supervised Learning Method for Text Chunking. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics*, pages 1–9.

Galen Andrew and Jianfeng Gao. 2007. Scalable Training of L1-regularized Log-linear Models. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 33–40. Omnipress.

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Improving Dependency Parsing with Subtrees from Auto-Parsed Data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 570–579.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113.

Gregory Druck and Andrew McCallum. 2010. High-Performance Semi-Supervised Learning using Discriminatively Constrained Generative Models. In *Proceedings of the International Conference on Machine Learning (ICML 2010)*, pages 319–326.

John Duchi and Yoram Singer. 2009. Efficient Online and Batch Learning Using Forward Backward Splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Terry Koo and Michael Collins. 2010. Efficient Third-Order Dependency Parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*, pages 595–603.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Dekang Lin and Xiaoyun Wu. 2009. Phrase Clustering for Discriminative Learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1030–1038.

Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Programming, Ser. B*, 45(3):503–528.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. 2010. Turbo Parsers: Dependency Parsing by Approximate Variational Inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44.

Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proceedings of ACL-08: HLT*, pages 665–673.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 551–560.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 477–485.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.