

Disambiguating named entities with deep supervised learning via crowd labels*

Le-kui ZHOU, Si-liang TANG, Jun XIAO, Fei WU[‡], Yue-ting ZHUANG

(Institute of Artificial Intelligence, College of Computer Science and Technology,
Zhejiang University, Hangzhou 310027, China)

E-mail: luckiezhou@zju.edu.cn; siliang@zju.edu.cn; junx@zju.edu.cn; wufei@zju.edu.cn; yzhuang@zju.edu.cn

Received Dec. 20, 2016; Revision accepted Dec. 22, 2016; Crosschecked Jan. 5, 2017

Abstract: Named entity disambiguation (NED) is the task of linking mentions of ambiguous entities to their referenced entities in a knowledge base such as Wikipedia. We propose an approach to effectively disentangle the discriminative features in the manner of collaborative utilization of collective wisdom (via human-labeled crowd labels) and deep learning (via human-generated data) for the NED task. In particular, we devise a crowd model to elicit the underlying features (crowd features) from crowd labels that indicate a matching candidate for each mention, and then use the crowd features to fine-tune a dynamic convolutional neural network (DCNN). The learned DCNN is employed to obtain deep crowd features to enhance traditional hand-crafted features for the NED task. The proposed method substantially benefits from the utilization of crowd knowledge (via crowd labels) into a generic deep learning for the NED task. Experimental analysis demonstrates that the proposed approach is superior to the traditional hand-crafted features when enough crowd labels are gathered.

Key words: Named entity disambiguation; Crowdsourcing; Deep learning

<http://dx.doi.org/10.1631/FITEE.1601835>

CLC number: TP391.4

1 Introduction

Named entities, which are uniquely identifiable objects with distinct existence, have now become one of the most important features on the Web. For example, the Linked Open Data (LOD) community has been publishing structured data for various applications based on entities; search engines nowadays exploit entities to enhance their search results (Haas *et al.*, 2011). However, there is still a large amount

of unstructured data on the Web, aiming at human consumption instead of easy machine understanding.

Linking the references of entities (a.k.a. surface form or mention) in these documents to entity-based knowledge is helpful for machines to understand semantics in unstructured documents, which also makes it possible for algorithms to enrich these documents by retrieving information from structured data. This task is called ‘named entity linking (NEL)’ or simply entity linking, and it is becoming an essential technology in the new artificial intelligence (AI) era (Pan, 2016).

Amongst entity linking tasks, named entity disambiguation (NED) is generally used to discern the true entity that each mention refers to, usually from a given candidate set. For example, NED algorithms attempt to distinguish the mention ‘apple’ between a kind of fruit and a technology company, depending

[‡] Corresponding author

* Project supported by the National Basic Research Program of China (No. 2015CB352300), the National Natural Science Foundation of China (Nos. 61402401 and U1509206), the Zhejiang Provincial Natural Science Foundation of China (No. LQ14F010004), the China Knowledge Centre for Engineering Sciences and Technology, the Fundamental Research Funds for the Central Universities, and the Qianjiang Talents Program of Zhejiang Province, China

 ORCID: Fei WU, <http://orcid.org/0000-0003-2139-8807>

© Zhejiang University and Springer-Verlag Berlin Heidelberg 2017

on its surrounding context.

In general, NED can be regarded as a classification task (e.g., classifying a given mention to an entity from a referenced knowledge base like Wikipedia), and the performance of entity disambiguation is heavily dependent on the generation of feature representations of the mention and the entity.

As shown in Fig. 1, conventional feature representations on NED are either hand-crafted features (e.g., term frequency-inverse document frequency (tf-idf) and bag-of-words (BoW)) or the learned features from raw data (sometimes labeled information is given). In our work, we consider learning feature representation from the perspective of optimizing the wisdom of crowds. Crowdsourcing techniques are employed to acquire sparse and noisy labels produced by human workers. Features learned from these crowd labels, which we call crowd features, are expected to reflect human interpretation of the task. Unlike conventional feature extraction methods, crowd features incorporate human knowledge into machine algorithms.

In this study, we are interested in the following problems: (1) How to effectively learn the crowd features from the wisdom of crowds (e.g., crowd labels)? (2) Are crowd features capable of enhancing hand-crafted features? (3) How to collaboratively use both crowd features and hand-crafted features to improve the NED performance by using traditional classification algorithms?

In particular, we propose an approach called deep supervised learning via crowd labels (DeCL). In DeCL, a crowd model is devised to elicit crowd features from crowd labels, and then the crowd features are employed to fine-tune a dynamic convolutional neural network (DCNN). The learned DCNN is employed to obtain deep crowd features to enhance the NED classification performance by traditional algorithms with the hand-crafted features. In online crowdsourced entity linking systems, the proposed DeCL yields the remarkable gain from crowdsourced labels by enhancing algorithm matchers and reducing the total cost in crowdsourcing.

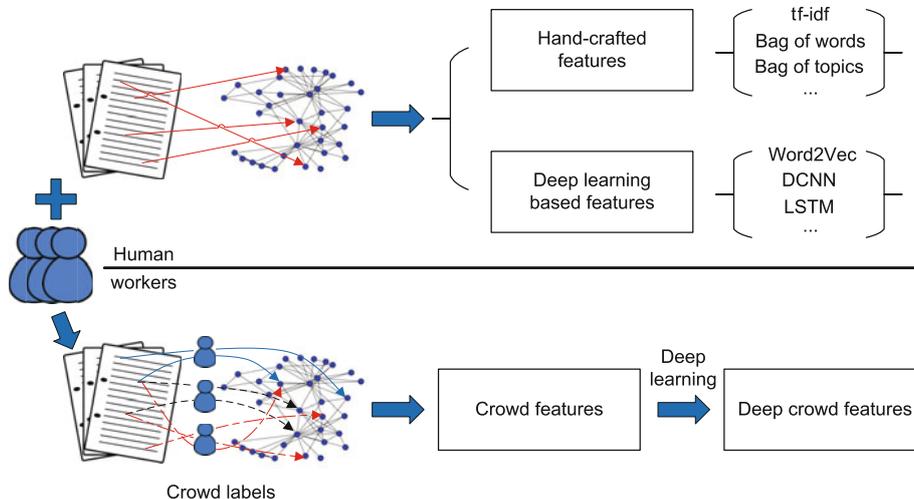


Fig. 1 Hand-crafted features widely employed in tasks in natural language processing. Typical hand-crafted features include term frequency-inverse document frequency (tf-idf) features and bag-of-words (BoW) features, which are both constructed according to a set of manually predefined rules. Other feature extraction algorithms, such as topic-based features (Blei *et al.*, 2003), employ a learning procedure. However, they still rely heavily on hand-designed a priori methods (e.g., a graph model). Neural network or deep learning based feature extraction methods, such as Word2Vec (Mikolov *et al.*, 2013), dynamic convolutional neural network (DCNN) (Kalchbrenner *et al.*, 2014), and long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), learn feature representations directly from the data itself, in either a supervised or an unsupervised manner. Both hand-crafted features and deep learning based methods are machine algorithms working on the data, and disregard the wisdom of the crowds. Our method, on the contrary, appropriately deals with non-perfect labels produced by human workers, and learns crowd features in a semi-supervised manner. Combining it with deep learning techniques, deep crowd features are finally learned in order to transfer the wisdom of crowds to various datasets

2 Related work

Various methods have been proposed for NEDs in recent years, and the most traditional methods of NEDs usually employ a vector space model (VSM). Bagga and Baldwin (1998) disambiguated entities across documents with a VSM-disambiguation module, which first extracts a concise summary for each entity and then discovers coreferences among the entities according to cosine similarities in terms of tf-idf features of the summaries. Some recent work used Wikipedia or DBpedia as an auxiliary source of information for NEDs. Gabrilovich and Markovitch (2007) proposed explicit semantic analysis (ESA), which first builds an inverted index of articles over Wikipedia, and then computes cosine similarity scores of articles using vectors built on the inverted index. Bunescu and Paşca (2006) employed cosine similarity between the mention context and Wikipedia articles, and trained a support vector machine (SVM) classifier based on the cosine similarity score and the taxonomy kernel. Cucerzan (2007) combined Wikipedia content with VSMS to handle large-scale NEDs, and an extended feature vector has been built based on the Wikipedia content and the entity is disambiguated according to a VSM.

Another principal class of NED methods uses a graph structure. Gentile *et al.* (2009) evaluated semantic relatedness over a graph built on the features extracted from Wikipedia. Hakimov *et al.* (2012) attempted to identify the correct entity by computing a centrality factor on the graph built from Wikipedia article links. Alhelbawy and Gaizauskas (2014) applied a ranking algorithm to an entity graph in order to rank the popularity of each candidate entity.

By introducing crowdsourcing techniques into NED tasks, ZenCrowd (Demartini *et al.*, 2012) dynamically generates micro-tasks on an online crowdsourcing platform and takes the advantage of human intelligence to improve the quality of the links. Moreover, a probabilistic framework is devised in ZenCrowd to identify unreliable human workers and make sensible decisions for entity linking, that is, inferring ground truth of entity linking from noisy labeling information by a crowd of non-experts.

Unlike ZenCrowd, in which crowdsourcing is used only as a tool to solve a specific problem, we consider crowdsourcing as one way to incorporate human intelligence into machine learning algorithms.

Specifically, we focus on whether it is possible to learn underlying features effectively (e.g., the crowd features in this study) from the crowd labels, so that human perspectives are properly incorporated into these features. We further discuss whether traditional classification algorithms benefit from these learned features.

3 Methods

The proposed DeCL takes the advantage of crowdsourced data and attempts to enhance the performance of mere machine-based NED algorithms.

The flowchart of the DeCL method is shown in Fig. 2. In DeCL, we first devise a crowd model which aggregates crowd labels and elicits underlying crowd features of all mentions and entities. Then we train a dynamic convolutional neural network (DCNN) fine-tuned by crowd features. The trained DCNN maps given mentions or entities to the so-called deep crowd features. In the end, the performance of the NED task is boosted by the collaborative utilization of deep crowd features and traditional hand-crafted features.

3.1 Notations

Assume that the set of mentions is $M = \{M_i\}_{i=1}^I$ and the set of entities is $E = \{E_j\}_{j=1}^J$. The goal of NED algorithms is to identify the true entity in E , with respect to a given mention M_i .

We denote the set of all involved workers by $U = \{U_k\}_{k=1}^K$. The crowdsourcing data can be formulated as a set of 4-tuples $\{(u_q, m_q, e_q, t_q)\}_{q=1}^Q$, where $u_q \in \{1, 2, \dots, K\}$ is the index of a worker in U , $m_q \in \{1, 2, \dots, I\}$ is the index of an involved mention, $e_q \subset \{1, 2, \dots, J\}$ is the set of candidate entity indices, and $t_q \in e_q$ is the index of the entity chosen by worker U_{u_q} . We further denote the set of all mention indices in crowdsourced data as $M_c = \cup_{p=1}^P \{m_p\}$, and the set of all entity indices in crowdsourced data as $E_c = \cup_{p=1}^P e_p$.

As discussed before, traditional to-be-improved algorithms also need a training set on NED tasks, with ground-truth labels provided. Note that in the training of traditional classifiers, they are not necessarily given any crowdsourced label.

The data for training traditional classifiers consists of a set of 3-tuples $\{(\hat{m}_p, \hat{e}_p, \hat{t}_p)\}_{p=1}^P$, where mention index $\hat{m}_p \in \{1, 2, \dots, I\}$, candidate entity

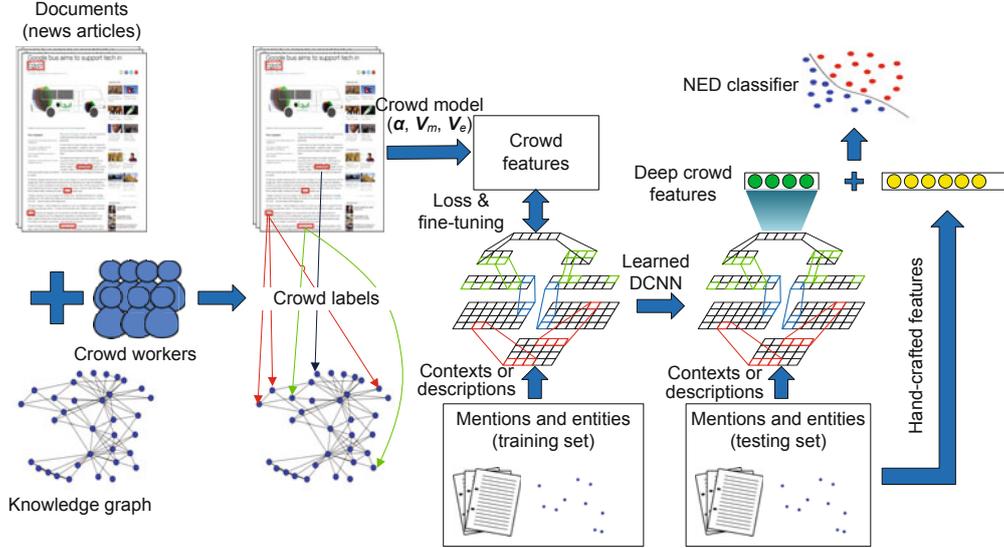


Fig. 2 Intuitive flowchart of the proposed DeCL. In DeCL, the crowd labels (indicating the linking between mentions in documents and entities in knowledge bases) are served as the input to a devised crowd model to elicit the underlying crowd features for mentions and entities. The crowd features are assumed to be compatible with the inferred true labels (e.g., the true linking). The crowd features are then used to fine-tune a dynamic convolutional neural network, which maps each mention of entities in the test data to deep crowd features. Finally, the deep crowd features and hand-crafted features are both employed to tackle NED tasks via an enhanced classifier

indices $\hat{e}_p \in \{1, 2, \dots, J\}$, and the ground-truth entity index $\hat{t}_p \in \hat{e}_p$.

3.2 Modeling the crowd

In crowdsourcing tasks, each problem is usually labeled multiple times by different workers for the sake of reliability. As a result, there might be conflicts among answers, and the labels are still not reliable enough due to the sparse and noisy nature of crowd labels. Unlike previous works on handling such ‘weakly labeled’ problems (Wu *et al.*, 2015), a crowd model is devised in our work for label aggregation. In this subsection, we describe a crowd model which embeds a d -dimensional vector (crowd features) to mentions or entities in accordance with the (noisy) crowd labels.

Using notation $\mathbf{v}(\cdot)$ as the corresponding d -dimensional crowd feature vector of a mention or an entity, then the matrices composed of crowd feature vectors are denoted as $\mathbf{V}_m = (\mathbf{v}_m(i) := \mathbf{v}(M_i))_{i=1}^I$ and $\mathbf{V}_e = (\mathbf{v}_e(j) := \mathbf{v}(E_j))_{j=1}^J$.

We introduce parameter $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_K)$ constrained by $\|\boldsymbol{\alpha}\|_2 = 1$ in order to resolve conflicts by modeling the ability of workers. Worker U_k always gives the answer in ‘common sense’ when

$\alpha_k = 1$ while the worker always gives the opposite answer when $\alpha_k = -1$, and the worker gives a random answer when $\alpha_k = 0$.

In our crowd model, $\boldsymbol{\alpha}$, \mathbf{V}_m , and \mathbf{V}_e are learned to minimize the loss function as follows:

$$L_1(\boldsymbol{\alpha}, \mathbf{V}_m, \mathbf{V}_e) = \sum_{q=1}^Q \sum_{j \in e_q} -\alpha_{u_q} \delta^*(j, t_q) \langle \mathbf{v}_e(j), \mathbf{v}_m(m_q) \rangle + \frac{\eta}{2} (\|\mathbf{v}_e(j)\|^2 + \|\mathbf{v}_m(m_q)\|^2) \quad (1)$$

s.t. $\|\boldsymbol{\alpha}\|_2 = 1,$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors, η is a regularization parameter, and $\delta^*(\cdot, \cdot)$ is the modified version of the Kronecker delta function given by

$$\delta^*(x, y) = \begin{cases} 1, & x = y, \\ -1, & x \neq y. \end{cases} \quad (2)$$

The value of the inner product $\langle \mathbf{v}_e(j), \mathbf{v}_m(m_q) \rangle$ measures the similarity between two crowd feature vectors. So, the first term in Eq. (1) can be translated as a penalty for the misclassifications taking into account users’ abilities. To be specific, the loss increases in proportion with a worker’s ability if a mention and an entity labeled by the worker to be

a pair of entity linking have dissimilar crowd feature vectors, and vice versa. The second term in Eq. (1) is a regularization term on the intensity of crowd feature vectors. Note that instead of adding the norms of \mathbf{V}_m and \mathbf{V}_e to the loss function, we regularize the loss function with the weighted sum of crowd feature vector norms, according to how many corresponding labels of each vector are provided by human workers.

We observe from Eq. (1) that, as the loss function is minimized, crowd features of a pair of a linked mention and an entity tend to be similar, where the similarity of two vectors is characterized by the large inner product between them, when more workers that are reliable mark them as a matched linking.

3.3 Learning from the crowd

As discussed before, discovering informative features is essential to the success of NED algorithms. The aforementioned crowd model is capable of embedding each entity and a mention in training data into d -dimensional crowd features by using crowd labels. Since the crowd features are compatible with human intelligence, it is attractive to employ the crowd features to refine a deep model, and then conduct the fine-tuned deep model to extract appropriate features from the data in the testing set.

As a result, we train a deep model discriminatively in a supervised fashion such that the distances between entities and mentions are minimized.

We denote the chosen deep model as γ with parameter θ . More specifically, a dynamic convolutional neural network (DCNN) described in Kalchbrenner *et al.* (2014) is employed in our framework. As shown in Fig. 3, the input of the DCNN is a feature matrix with each column corresponding to a word in a textual paragraph, and the output of the DCNN is a d -dimensional vector. Apart from the input layer (a feature matrix) as well as the fully connected layer preceding the output, the remaining parts of the DCNN are composed of repetitive hidden layers with identical meta-structure. A hidden layer can be further divided into four sublayers, namely a convolution sublayer, a dynamic k -max pooling sublayer, a folding sublayer, and a nonlinear sublayer. All these sublayers except the nonlinear sublayer are illustrated in Fig. 3.

We extract the textual contexts from mentions and entities, and represent each word in the textual paragraph with a real vector using the method pro-

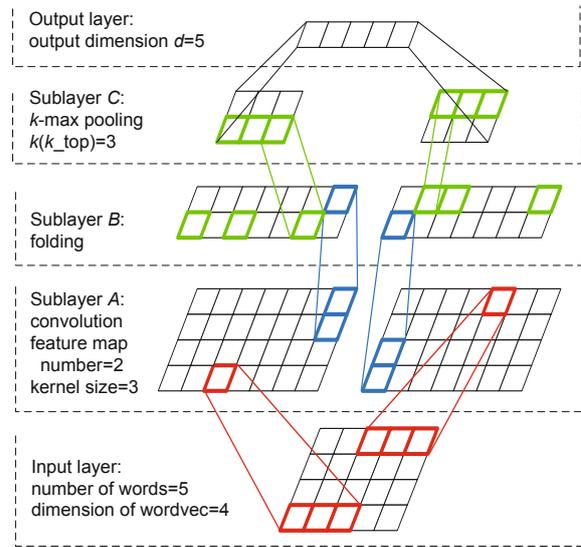


Fig. 3 Structure of the DCNN. We show a DCNN with five words as the input and each word is represented as a vector of four dimensions. The output of the DCNN is a 5-dimensional vector. The network has one middle layer involved, which consists of sublayers A, B, and C. Convolution sublayer A contains two feature maps, both of which are obtained by the convolution of the input feature matrix with size-3 kernels. Sublayer B is the folding sublayer, in which adjacent rows of feature maps are combined. Sublayer C is a k -max pooling layer, in which we select three maximum elements in each row from feature maps in the previous sublayer

posed in Mikolov *et al.* (2013). For each textual paragraph, we combine the word vectors by placing them sequentially into columns of a matrix. We denote the feature matrices by $\mathbf{s}_m(i)$ and $\mathbf{s}_e(j)$ for mention M_i and entity E_j , respectively. Finally, we can obtain a feature matrix for each mention or entity in the crowdsourcing data, denoted by $\mathbf{S}_m = (\mathbf{s}_m(i))_{i \in M_c}$ for mentions M_c and $\mathbf{S}_e = (\mathbf{s}_e(j))_{j \in E_c}$ for entities E_c . Note that in practice, the embedded vectors of words that we use to build the feature matrices are obtained from pretrained results on part of the Google News dataset, provided by the Word2Vec Project (<https://code.google.com/p/word2vec/>).

We fine-tune the DCNN with a loss function based on the Euclidean distances as follows:

$$L_2(\theta) = \sum_{i \in M_c} \|\gamma(\mathbf{s}_m(i); \theta) - \mathbf{v}_m(i)\|^2 + \sum_{j \in E_c} \|\gamma(\mathbf{s}_e(j); \theta) - \mathbf{v}_e(j)\|^2 + \xi \|\theta\|^2, \quad (3)$$

where ξ is a regularization parameter. The above can be solved in practice by backpropagating the gradient of the loss function and applying regularization techniques such as a weight decay. By leveraging crowd features (the wisdom of the crowd), the DCNN can be optimized directly to disentangle the underlying features of mentions and entities. We call the feature vectors output by the fine-tuned DCNN $\gamma(\cdot; \theta)$ deep crowd features.

3.4 Boosting a classifier with deep crowd features

We argue that a classifier for the NED task can substantially benefit from the collaborative utilization of deep crowd features and traditional hand-crafted features.

Now that we have trained a DCNN $\gamma(\cdot; \theta)$, for each mention M_i and each entity E_j , and we are able to create feature matrices $\mathbf{s}_m(i)$ and $\mathbf{s}_e(j)$ from their textual contexts, as described in the previous subsection. These feature matrices are transformed by the learned DCNN as $\mathbf{v}_m(i) = N(\mathbf{s}_m(i); \theta)$ and $\mathbf{v}_e(j) = N(\mathbf{s}_e(j); \theta)$ for mention M_i and entity E_j , respectively. These are what we call deep crowd feature vectors since the DCNN is trained with the help of crowd features.

For each mention M_i and each entity E_j , we denote their hand-crafted feature vectors as $\mathbf{w}_m(i)$ and $\mathbf{w}_e(j)$, respectively. Then we concatenate their hand-crafted features and deep crowd features as follows:

$$\mathbf{w}_m^*(i) = \mathbf{w}_m(i) \frown \mathbf{v}_m(i), \quad (4)$$

$$\mathbf{w}_e^*(j) = \mathbf{w}_e(j) \frown \mathbf{v}_e(j), \quad (5)$$

where symbol ' \frown ' is for vector concatenation. After giving a mention, the concatenation of its deep crowd features (via the fine-tuned DCNN) and hand-crafted features are fed into a classifier to perform NED.

3.5 Classification-based NED algorithms

In the final step of our algorithm pipeline, we feed our combined features to a classification-based NED algorithm, which will be discussed in this subsection in a little more detail.

Classification-based methods have been adopted in NED problems in order to handle heterogeneous features (Bunescu and Paşca, 2006), where an NED

task is transformed into a classification problem, which can be solved effectively by standard classifiers. With a slight abuse of notation, we denote general feature vectors of mention M_i and entity E_j by $\mathbf{w}_m(i)$ and $\mathbf{w}_e(j)$ respectively in this subsection.

First, we define a joint feature function for arbitrary feature vectors \mathbf{x} and \mathbf{y} by

$$\mathbf{J}(\mathbf{x}, \mathbf{y}) = |\mathbf{x} - \mathbf{y}|, \quad (6)$$

where $|\cdot|$ is an element-wise absolute function. Given each 3-tuple $(\hat{m}_p, \hat{e}_p, \hat{t}_p)$ in the training set, we generate training features and the corresponding labels by

$$\mathbf{w}_p = \{(\mathbf{J}(\mathbf{w}_m(\hat{m}_p), \mathbf{w}_e(j)), \delta^*(j, \hat{t}_p))\}_{j \in \hat{e}_p}, \quad (7)$$

where $\delta^*(\cdot, \cdot)$ is the modified Kronecker delta function described in Eq. (2).

For a general classification algorithm that outputs both class labels and confidence scores, we have it trained on generated data $W = \cup_{p=1}^P \mathbf{w}_p$. Then for any pair (M_i, E_j) , it is possible for us to define a similarity score $S(M_i, E_j)$ and a class label $L(M_i, E_j)$, based on the confidence score and the class label we obtained by applying the classifier on feature vector $\mathbf{J}(\mathbf{w}_m(i), \mathbf{w}_e(j))$. According to how we generate the training data, a positive $L(M_i, E_j)$ indicates that E_j matches M_i while a negative label indicates a mismatch.

Given any mention $M_i \in M$, we compute similarity scores $S(M_i, E_j) \forall j \in E$ and pick the top-scoring entity as the correct match of M_i . In case we are expected to choose nothing if all candidates mismatch, we reject all candidates if all labels $L(M_i, E_j) (\forall j \in E)$ are negative.

We choose AdaBoost as the classification algorithm in our experiments, where the confidence scores and class labels are obtained intuitively according to the intensity and the sign of ensemble weak classifier scores.

4 Optimization

4.1 Inferencing the crowd model

It is difficult to solve our crowd model directly by the Lagrange multipliers. Instead, we approximate the optimization of the crowd model in a two-step scheme.

In the first step, we fix the value of parameter α and optimize \mathbf{V}_e and \mathbf{V}_m using the stochastic gradient descent as follows:

$$\mathbf{V}_m^{(t+1)} = \mathbf{V}_m^{(t)} + \epsilon_1 \cdot \mathbf{g}_m^{(t)}, \quad (8)$$

$$\mathbf{V}_e^{(t+1)} = \mathbf{V}_e^{(t)} + \epsilon_1 \cdot \mathbf{g}_e^{(t)}, \quad (9)$$

where ϵ_1 is the learning rate, $\mathbf{g}_m^{(t)}$ and $\mathbf{g}_e^{(t)}$ are the gradients of a single summation term in Eq. (1) with respect to \mathbf{V}_m or \mathbf{V}_e at step t . The i th element in $\mathbf{g}_m^{(t)}$ and the j th element in $\mathbf{g}_e^{(t)}$ are given as follows:

$$(g_m^{(t)})_i = \delta(m_q, i) \left(\sum_{j \in e_q} -\alpha_{u_q} \delta^*(j, t_q) \mathbf{v}_e^{(t)}(j) + \eta \mathbf{v}_m^{(t)}(m_q) \right), \quad (10)$$

$$(g_e^{(t)})_j = \delta(i, j) \left(\sum_{j \in e_q} -\alpha_{u_q} \delta^*(j, t_q) \mathbf{v}_m^{(t)}(m_q) + \eta \mathbf{v}_e^{(t)}(j) \right), \quad (11)$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta function and $\delta^*(\cdot, \cdot)$ is its modified version defined by Eq. (2).

In the second step, we fix \mathbf{V}_m and \mathbf{V}_e , and optimize α , which is a simple problem of optimizing a linear objective function with an equality constraint. We first simplify the notation of the problem as follows:

$$L_1(\alpha) = \sum_{k=1}^K C_k \alpha_k + C_0 \quad \text{s.t.} \quad \|\alpha\| = 1, \quad (12)$$

where

$$C_0 = \sum_{q=1}^Q \sum_{j \in e_q} \frac{\eta}{2} (\|\mathbf{v}_e(j)\|^2 + \|\mathbf{v}_m(m_q)\|^2), \quad (13)$$

$$C_k = \sum_{q=1}^Q \sum_{j \in e_q} -\delta(u_q, k) \delta^*(j, t_q) \langle \mathbf{v}_e(j), \mathbf{v}_m(m_q) \rangle, \quad \forall k \in \{1, 2, \dots, K\}. \quad (14)$$

Eq. (12) can be effectively solved with the Lagrange multipliers, and the result is given by

$$\alpha_k = \frac{C_k}{\sqrt{C_1^2 + C_2^2 + \dots + C_K^2}}, \quad \forall k \in \{1, 2, \dots, K\}, \quad (15)$$

or

$$\alpha_k = \frac{-C_k}{\sqrt{C_1^2 + C_2^2 + \dots + C_K^2}}, \quad \forall k \in \{1, 2, \dots, K\}, \quad (16)$$

whichever produces a lower loss.

4.2 Fine-tuning DCNN

We followed exactly the same way as in Kalchbrenner *et al.* (2014) to train the DCNN, except that we use the loss function in Eq. (3) for fine-tuning. To show the way our loss function fits into the stochastic gradient descent framework, we first define a set of 2-tuples $R = \{(i, 0)\}_{i \in M_c} \cup \{(0, j)\}_{j \in E_c}$ and the number of tuples in R is $|M_c| + |E_c|$. Then we can rewrite Eq. (3) as

$$L_2^*(\theta) = \sum_{(i,j) \in R} \|\gamma(\mathbf{s}_*(i, j); \theta) - \mathbf{v}_*(i, j)\|^2, \quad (17)$$

where

$$\mathbf{s}_*(i, j) = \begin{cases} \mathbf{s}_m(i), & i > 0 \wedge j = 0, \\ \mathbf{s}_e(j), & i = 0 \wedge j > 0, \end{cases} \quad (18)$$

$$\mathbf{v}_*(i, j) = \begin{cases} \mathbf{v}_m(i), & i > 0 \wedge j = 0, \\ \mathbf{v}_e(j), & i = 0 \wedge j > 0. \end{cases} \quad (19)$$

Note that the regularization term on θ is ignored here because it is brought in by neural network tuning techniques such as weight decay in practice.

Now Eq. (17) can be solved by applying the stochastic gradient descent. When given $(i, j) \in \mathbb{R}$, θ is updated by

$$\theta^{(t+1)} = \theta^{(t)} + \epsilon_2 \cdot \mathbf{g}_\theta^{(t)}, \quad (20)$$

where ϵ_2 is the learning rate and

$$\mathbf{g}_\theta^{(t)} = 2(\gamma(\mathbf{s}_*(i, j); \theta^{(t)}) - \mathbf{v}_*(i, j)) \left. \frac{\partial \gamma(\mathbf{s}_*(i, j); \theta)}{\partial \theta} \right|_{\theta=\theta^{(t)}}. \quad (21)$$

The derivative part in the last equation can be easily obtained via back propagation on DCNN.

5 Experiments and results

5.1 Dataset

We train and test our algorithm on the dataset provided in Demartini *et al.* (2012), which is a set

of Human Intelligence Tasks (HITs) records generated by Amazon Mechanical Turk (AMT) users. In these HITs, human workers link from mentions detected in several news articles to four knowledge bases, namely DBpedia (<http://www.dbpedia.org>), FreeBase (<http://www.freebase.com>), GeoNames (<http://www.geonames.org>), and New York Times (<http://data.nytimes.com>). The ground-truth labels for training baseline algorithms are provided by a group of authorized users, who will be treated as normal users when we learn the crowdsourced features.

The dataset is first pre-processed by discarding unavailable online news articles, filtering out broken entity URLs and eliminating invalid HITs. The statistics of the remaining part of the dataset are shown in Table 1. There are 487 mentions involved in this dataset, and 1669 entities are extracted as candidates from the four knowledge bases.

Table 1 Statistics of the ZenCrowd dataset

Attribute	Statistics		Attribute	Statistics
Mention	487		User	84
Knowledge base	4		Label	7691
Candidate entity	1669		Average	91.56
NED task	1050		labels/user	

When we link a single entity to different knowledge bases, these can be treated as distinct NED tasks. As a result, 1050 valid NED tasks are available in this dataset. These tasks are labeled by 84 workers, producing 7691 links from mentions to candidate entities. Note that it is also possible that a user rejects all candidate entities for a mention, in which case we assign a ‘generalized label’ linking the mention to an empty virtual candidate. There are 11 268 generalized labels in total, and 134.14 per user on average.

We select a number of these tasks as the training data while the rest are treated as the testing data. The ratio of training-testing data size remains a variable in our experiments. We also split the crowdsourced labels according to whether the labeled task is in the training set. Generally speaking, a larger training set of NED tasks indicates more crowd labels for us to train our model.

Evidence can be found in this dataset that human and machine algorithms do not always agree with each other. Here, two examples are listed in Table 2, showing that there are cases for both crowd

workers and algorithms to outperform each other.

We can see in the first sample that the mention ‘Rhode Island Avenue’ is incorrectly linked by the algorithm to an avenue, while two out of three workers label it correctly as a station. In the second sample, the algorithm correctly identifies the ‘United States’ as a nation, and most of the workers give a wrong result. It can be concluded from the above examples that humans sometimes view the problem differently from a machine algorithm. The power of crowd features rises from these differences, such that they are expected to be good complements of conventional features.

5.2 Processing and configuration

For each mention, we extract the context surrounding its first appearance in the news articles with the window size set to 50. As for each entity, we extract a descriptive paragraph in English from the knowledge base as its description.

Hand-crafted features such as tf-idf features are then extracted from mention contexts and entity descriptions. To introduce these features in brief, a tf-idf feature is a widely used text feature weighing each word in the vocabulary by the product of its term frequency in a document and its inverse document frequency on a corpus.

In our experiments, the length of deep crowd features is set to $d = 5$ or 300, and the regularization parameter $\eta = 0.5$ (we do not care about ξ since regularization of DCNN is achieved by some tuning techniques in practice). We use a three-layer DCNN, with 3, 4, 2 feature maps and 6, 5, 3 convolution kernel sizes in each layer in the order from input to output. We employ an AdaBoost classifier with decision trees as weak learners.

5.3 Experimental results

Table 3 shows the NED accuracies of our algorithm on the ZenCrowd dataset. We can see from the results that the accuracy on combined features outperforms that on tf-idf features alone in all cases except when the ratio of training-testing data size is 5:5. The more crowd labels we use to train our model, the larger performance gain we obtain with the DeCL method.

Also, note that the great variance in deep crowd feature dimensions between 5 and 300 does not make

Table 2 Two samples from the ZenCrowd dataset

Mention detected	User ID / Algorithms	Candidate entities	Mention detected	User ID / Algorithms	Candidate entities
... trains will share a track between New York Avenue and Rhode Island Avenue stations ...	33	RIA (station)^a	... they planned to work closely with the United States , Europe and India to plan ...	7	These United States (rock band) ^c
	26	RIA (station)^a		59	Electoral College (institution) ^d
	47	RIA (avenue) ^b		39	United States (state)^e
	tf-idf + CosDistance	RIA (avenue) ^b		tf-idf + CosDistance	United States (state)^e

In the first columns, there are pieces of sentences in the news articles, in which the mention is marked in bold font. The second columns indicate whether the label is produced by a user or an algorithm, and user IDs or the algorithm descriptions are presented correspondingly. The third columns specify the entity that users or algorithms link the mention to, with correct links marked in bold font. Note that users are allowed to reject all candidate entities in this dataset, and such cases are not shown in this table due to the limited space

^a http://dbpedia.org/page/Rhode_Island_Avenue_%E2%80%93Brentwood_%28WMATA_station%29

^b http://dbpedia.org/page/Rhode_Island_Avenue_%28Washington,_D.C.%29

^c http://dbpedia.org/page/These_United_States

^d http://dbpedia.org/page/Electoral_College_%28United_States%29

^e http://dbpedia.org/page/United_States

Table 3 Accuracy comparisons on NED tasks in terms of two dimensions of deep crowd features

Feature	Accuracy							
	Deep crowd feature dimension: 5				Deep crowd feature dimension: 300			
	9:1	8:2	7:3	9:1	8:2	7:3	5:5	
tf-idf + DCF	78%	71%	69%	77%	71%	69%	67%	
tf-idf	74%	70%	69%	74%	70%	69%	69%	

Accuracies of NED tasks are given by applying AdaBoost on combined features (tf-idf + deep crowd features (DCF)) and tf-idf features respectively, in the settings of different ratios (9:1, 8:2, 7:3, and 5:5) of training data size to testing data size. tf-idf dimensions are the same, i.e., 18 478

much difference in NED accuracy, which we think is because they carry similar amount of information transferred from the crowd labels, for which even a 5-dimensional vector is expressive enough.

6 Conclusions and future work

We conclude from our experimental results that our methods can effectively improve conventional vector space NED algorithms even with a compact auxiliary feature space. We believe such enhancement to be more significant with more crowd labels involved, according to our observation on the effects of different ratios of training-testing data sizes. We believe this to be a promising way to make full use of crowdsourced labels produced by NED/NEL systems that arm themselves with the power of the crowd.

However, there is still much room for improvement: (1) It is beneficial to devise a more subtle crowd model and make full use of information from

HIT records; (2) An efficient online training scheme is needed to incorporate our method into an online NEL system.

References

- Alhelbawy, A., Gaizauskas, R., 2014. Graph ranking for collective named entity disambiguation. Proc. 52nd Annual Meeting of the Association for Computational Linguistics, p.75-80.
- Bagga, A., Baldwin, B., 1998. Entity-based cross-document coreferencing using the vector space model. Proc. 36th Annual Meeting of the Association for Computational Linguistics and 17th Int. Conf. on Computational Linguistics, p.79-85. <http://dx.doi.org/10.3115/980845.980859>
- Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, **3**:993-1022.
- Bunescu, R., Paşca, M., 2006. Using encyclopedic knowledge for named entity disambiguation. Proc. 11th Conf. of the European Chapter of the Association for Computational Linguistics, p.3-7.
- Cucerzan, S., 2007. Large-scale named entity disambiguation based on Wikipedia data. Proc. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, p.708-716.

- Demartini, G., Difallah, D.E., Cudré-Mauroux, P., 2012. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. Proc. 21st Int. Conf. on World Wide Web, p.469-478. <http://dx.doi.org/10.1145/2187836.2187900>
- Gabrilovich, E., Markovitch, S., 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. Proc. Int. Joint Conf. on Artificial Intelligence, p.1606-1611. <http://dx.doi.org/10.1145/2063576.2063865>
- Gentile, A.L., Zhang, Z.Q., Xia, L., et al., 2009. Graph-based semantic relatedness for named entity disambiguation. Proc. Int. Conf. on Software, Services & Semantic Technologies, p.13-20.
- Haas, K., Mika, P., Tarjan, P., et al., 2011. Enhanced results for web search. Proc. 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.725-734. <http://dx.doi.org/10.1145/2009916.2010014>
- Hakimov, S., Oto, S.A., Dogdu, E., 2012. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. Proc. 4th Int. Workshop on Semantic Web Information Management, Article 4. <http://dx.doi.org/10.1145/2237867.2237871>
- Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. *Neur. Comput.*, **9**(8):1735-1780.
- Kalchbrenner, N., Grefenstette, E., Blunsom, P., 2014. A convolutional neural network for modelling sentences. Proc. 52nd Annual Meeting of the Association for Computational Linguistics, p.655-665.
- Mikolov, T., Sutskever, I., Chen, K., et al., 2013. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, p.3111-3119.
- Pan, Y.H., 2016. Heading toward artificial intelligence 2.0. *Engineering*, **2**(4):409-413. <http://dx.doi.org/10.1016/J.ENG.2016.04.018>
- Wu, F., Wang, Z.H., Zhang, Z.F., et al., 2015. Weakly semi-supervised deep learning for multi-label image annotation. *IEEE Trans. Big Data*, **1**(3):109-122. <http://dx.doi.org/10.1109/TBDDATA.2015.2497270>