Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Sembler: Ensembling Crowd Sequential Labeling for Improved Quality

Xian Wu
Shanghai Jiao Tong University
Shanghai, 200240, China
wuxian@apex.sjtu.edu.cn

Wei Fan
IBM T.J.Watson Research Center
Yorktown Heights, New York
weifan@us.ibm.com

Yong Yu
Shanghai Jiao Tong University
Shanghai, 200240, China
yyu@apex.sjtu.edu.cn

Abstract

Many natural language processing tasks, such as named entity recognition (NER), part of speech (POS) tagging, word segmentation, and etc., can be formulated as sequential data labeling problems. Building a sound labeler requires very large number of correctly labeled training examples, which may not always be possible. On the other hand, crowdsourcing provides an inexpensive yet efficient alternative to collect manual sequential labeling from non-experts. However the quality of crowd labeling cannot be guaranteed, and three kinds of errors are typical: (1) incorrect annotations due to lack of expertise (e.g., labeling gene names from plain text requires corresponding domain knowledge); (2) ignored or omitted annotations due to carelessness or low confidence; (3) noisy annotations due to cheating or vandalism. To correct these mistakes, we present Sembler, a statistical model for ensembling crowd sequential labelings. Sembler considers three types of statistical information: (1) the majority agreement that proves the correctness of an annotation; (2) correct annotation that improves the credibility of the corresponding annotator; (3) correct annotation that enhances the correctness of other annotations which share similar linguistic or contextual features. We evaluate the proposed model on a real Twitter and a synthetical biological data set, and find that Sembler is particularly accurate when more than half of annotators make mistakes.

Introduction

Statistical models have been applied with great success to a wide range of natural language processing tasks, such as CRF (Conditional Random Fields) for named entity recognition (NER) and SVM (Support Vector Machine) for polarity classification. Building a sound statistical model requires sufficient number of manual annotations, which is typically quite large. It is expensive and time consuming to obtain these labeled examples, and may not always be possible for many real-world applications. The recent flourishing of Crowdsourcing services, such as Amazon Mechanical Turk (MTurk) [1] and CrowdFlower [2] , provide an inexpensive and efficient method to acquire manual annotations. Snow et al. (2008) has published five NLP annotation tasks on MTurk and demonstrated the effectiveness and efficiency of crowd annotation on these tasks.

However the annotators in Crowdsourcing services are ordinary web users, and their devotions and expertise are not evaluated in advance. Therefore the quality of crowd annotations cannot be guaranteed. Various errors caused by the lack of expertise, inexperience, carelessness, vandalism and etc. are quite common. To remove these errors and preserve high quality annotations, some refinement approaches have been proposed. Snow et al. (2008) provides a bias correction method for combining non-expert annotations; Dempster, Laird, and Rubin (1977) introduces a statistical framework to merge noisy labels from multiple sources and infer the correct one; Raykar et al. (2009) further integrates linear regression model into the statistical framework proposed by (Dempster, Laird, and Rubin 1977).

For these previously proposed approaches, it is assumed that the instances to be annotated are independent from each other. And each instance is viewed as an atomic unit where the internal dependencies is not considered. However, for many natural language processing tasks, like named entity recognition, POS tagging and word segmentation, the input instances are “sequential data” or sequences of words/entities, where statistical dependencies exists not only among instances but also among entities inside instances. For example, in a named entity recognition task, the annotator is required to label the appearances of named entities in sentences. The annotator needs to provide a reasonable label sequence for the whole sentence instead of labeling each token in each sentence separately from the context. In other words, for sequential labeling task, the typical *i.i.d.* assumption is no longer true.

To cope with noisy and potentially erroneous crowd annotations for sequential data, we provide Sembler, a statistical model to ensemble or combine crowd sequential labelings. Similar to (Snow et al. 2008) and (Raykar et al. 2009), we consider the majority agreement and the annotator’s credibility in developing this model. As an import improvement over previous work, we also model the relationships/dependencies in sequential data.

For example, in the gene name annotation task given below, Sentence 1 is correctly annotated by Annotator 1, as the

[1]https://www.mturk.com/mturk/welcome

[2]http://crowdflower.com/

appearance of gene name "FGF1" is labeled; while the labeling on Sentence 2 is wrong, as the Annotator 2 missed the gene name "FGF2". From the first correct annotation, Sembler can learn three clues to assign the label ***B*** to a token: (1) has the prefix "FGF"; (2) is surrounded by words "activation" and "by"; (3) the previous token is labeled as ***O***. Combining these three clues, Sembler can correct the mistake made by annotator 2 and assign the label ***B*** to "FGF2". In this manner, Sembler takes advantage of the correlation and dependency among multiple instances to improve performance, and this statistical correlation and dependency is ignored by state-of-the-art methods.

Sentence 1:	activation	induced	by	FGF1
Annotator 1:	***O***	***O***	***O***	***B***
Sentence 2:	promoter	activation	by	FGF2
Annotator 2:	***O***	***O***	***O***	***O***

In order to enable such functions in Sembler, we need to model the dependencies among instances, as well as the rich set of local features (e.g., prefix, adjacent words, etc) of sequential data. To do so, we extend the statistical framework of CRF and propose a parametric model to simulate the generating process of crowd sequential labelings. By maximizing the likelihood (or MLE), the parameters involved in this model can be inferred. With these parameters, we can either (1) generate refined and unified sequential labelings from noisy and erroneous crowd inputs or (2) provide a sequential data annotator directly. In other words, Sembler is able to train a unified sequential annotator from crowd sequential labelings. We demonstrate the advantages of Sembler over majority vote on a Twitter corpus and a synthetic biological data set. Most importantly, we find that (1) Sembler have higher accuracy ranging from about 5% to over 10%, when the quality of crowd sequential labeling is quite poor (or most crowd answers are wrong), and (2) the margin of improvement actually grows larger as the quality of crowd sequential labeling is poorer.

The Framework

Problem Definition

Definition 1 (Sequential Data) *We define a sequence of entities to be an instance of sequential data. In the context of named entity recognition, each entity represents a single word, an instance of sequential data is thus a word sequence* $(w_1, w_2, \ldots, w_M)$*. As follows, we use* $X^{(i)}$ *to denote an instance and* $\mathcal{X} = \{X^{(1)}, X^{(2)}, \ldots, X^{(N)}\}$ *to denote the set of sequential data.*

Definition 2 (Sequential Labeling) *A sequence labeling is the assignment of labels on each entity of the sequential data. For example, in NER tasks, the word sequence* $(w_1, w_2, \ldots, w_M)$ *is labeled with* $(l_1, l_2, \ldots, l_M)$*, where* l_i *is the label assigned for word* w_i*. Each label is selected from an alphabet set, typically, {B,I,O}, to denote the* **B***eginning, the* **I***ntermediate and the non-named entity tokens or* **O***bjects.*

Definition 3 (Crowd Sequential Labeling) *The crowd sequence labeling are conducted by multiple annotators from crowdsourcing services. Let* $\{A_1^{(i)}, A_2^{(i)}, \ldots, A_K^{(i)}\}$ *denote the crowd sequential labeling conducted on instance* $X^{(i)}$ *by* K *different annotators.* $A_k^{(i)}$ *is a sequential labeling conducted by the* k^{th} *annotator.*

Symbol	Description
$\mathcal{X}$	set of sequential data
$\mathcal{A}$	set of crowd sequential labeling on $\mathcal{X}$
$X^{(i)}$	i^{th} instance in $\mathcal{X}$
$A_k^{(i)}$	crowd sequential labeling on $X^{(i)}$ conducted by the k^{th} annotator
$\{A_k^{(i)}\}$	set of all crowd sequential labeling on $X^{(i)}$, short form of $\{A_1^{(i)}, A_2^{(i)}, \ldots, A_K^{(i)}\}$.
$L_t^{(i)}$	t^{th} possible sequential labeling on $X^{(i)}$
$\{L_t^{(i)}\}$	set of all possible sequential labeling on $X^{(i)}$, short form of $\{L_1^{(i)}, L_2^{(i)}, \ldots\}$.
N	number of instances in $\mathcal{X}$
K	number of annotators
x_{ij}	j^{th} entity in the instance $X^{(i)}$
a_{ijk}	label on j^{th} entity of $X^{(i)}$ in the labeling sequence $A_k^{(i)}$
l_{ijt}	label on j^{th} entity of $X^{(i)}$ in the labeling sequence $L_t^{(i)}$

Table 1: Notation Used in this Paper

As follows, we present a parametric framework to model the conditional probability distribution Pr$(\mathcal{A}|\mathcal{X})$, where $\mathcal{X}$ is the set of sequential data and $\mathcal{A}$ represents the crowd sequential labeling on $\mathcal{X}$. Please note that, here we model the conditional probability distribution Pr$(\mathcal{A}|\mathcal{X})$ instead of the joint probability distribution Pr$(\mathcal{A}, \mathcal{X})$. This is because modeling the joint probability distribution requires to model the distribution Pr$(\mathcal{X})$, which involves non-trivial statistical dependencies. Besides, the conditional probability distribution is sufficient for sequential data labeling.

The proposed approach is to formulate Pr$(\mathcal{A}|\mathcal{X})$ as parametric models, and then solve it by maximizing its likelihood on crowd sequential labelings, in order to obtain values of parameters. As shown later in this paper, we can use these parameters to predict the labels of unseen sequential data. Interestingly, if we re-label the training data with these parameters, the sequential labelings from multiple annotators can be merged into one unified labeling. As a result, many existing sequential data models, such as HMM (Hidden Markov Model) (Rabiner 1990) MeMM (Maximum Entropy Markov Model) (McCallum, Freitag, and Pereira 2000) and CRF (Conditional Random Fields) (Lafferty, McCallum, and Pereira 2001), that are initially designed for single source training data, can be applied to crowd sequential labeling. The notations used in this paper are summarized in Table 1.

The Ensembling Model: Sembler

We represent the conditional probability Pr$(\mathcal{A}|\mathcal{X})$ in Eq.(1).

$$\begin{aligned}\Pr(\mathcal{A}|\mathcal{X}) &= \prod_{i=1}^{N} \Pr(\{A_k^{(i)}\}|X^{(i)}) \\ &= \prod_{i=1}^{N}\sum_t \Pr(\{A_k^{(i)}\}|L_t^{(i)}, X^{(i)})\Pr(L_t^{(i)}|X^{(i)}) \\ &= \prod_{i=1}^{N}\sum_t \prod_{k=1}^{K} \Pr(A_k^{(i)}|L_t^{(i)})\Pr(L_t^{(i)}|X^{(i)}) \quad (1)\end{aligned}$$

where $L_t^{(i)}$ represents one possible sequential labeling on $X^{(i)}$ and the set $\{L_1^{(i)}, L_2^{(i)}, \ldots\}$ represents all possible sequential labeling on $X^{(i)}$. In Eq.(1), we assume the value of $A_k^{(i)}$ only relies on $L_t^{(i)}$ and is independent from $X^{(i)}$. Thus we have:

$$\Pr(\{A_k^{(i)}\}|L_t^{(i)}, X^{(i)}) = \Pr(\{A_k^{(i)}\}|L_t^{(i)})$$

In Eq.(1), $\Pr(\mathcal{A}|\mathcal{X})$ is decomposed into the combinations of two simpler probabilities:

1. $\Pr(L_t^{(i)}|X^{(i)})$ denotes the conditional probability of assigning the labels $L_t^{(i)}$ to the instance $X^{(i)}$.
2. $\Pr(A_k^{(i)}|L_t^{(i)})$ denotes the probability of transforming the labeling on instance $X^{(i)}$ from $L_t^{(i)}$ to $A_k^{(i)}$.

Using parametric modeling, we then introduce two sets of parameters ϑ and φ to model these two probabilities separately.

Since $\Pr(L_t^{(i)}|X^{(i)})$ does not involve crowd related variables, we can adopt existing methods designed for single-source labeling to estimate this probability. For example, in the framework of CRF, the statistical dependencies between $L_t^{(i)}$ and $X^{(i)}$ are modeled by an undirected graph (which we simplify into a chain representation in this paper) and the conditional probability $\Pr(L_t^{(i)}|X^{(i)})$ can be represented in Eq. (2).

$$\Pr(L_t^{(i)}|X^{(i)}) = \frac{\exp\left(\sum_{j,p}\lambda_p f_p(l_{i(j-1)t}, l_{ijt}, x_{ij})\right)}{Z(X^{(i)})} \quad (2)$$

where $\vartheta = (\lambda_1, \lambda_2, \ldots, \lambda_P)$ is the set of parameters to be estimated, and $Z(X^{(i)})$ is an instance specific normalization function.

$$Z(X^{(i)}) = \sum_t \exp\left(\sum_{j,p}\lambda_p f_p(l_{i(j-1)t}, l_{ijt}, x_{ij})\right) \quad (3)$$

Since the feature function f_p is common to all instances in sequential data. The total number of parameters in ϑ is equal to the size of feature functions, or P.

Then the problem is to formulate $\Pr(A_k^{(i)}|L_t^{(i)})$ with parametric modeling. Since the annotators vary in expertise and responsibility, we need to introduce parameters to model the performance of each individual annotator. Here we extend the parameters proposed by (Raykar et al. 2009) and use $\varphi = (\alpha, \beta, \gamma)$ to calculate the probability $\Pr(A_k^{(i)}|L_t^{(i)})$.

We exemplify the meaning of φ in an NER task. Assume the adopted label alphabet is $\{$***B, I, O***$\}$ where ***B*** represents the beginning token of a named entity, ***I*** represents the intermediate token of a named entity and ***O*** represents other tokens. An NER process is to assign each token in text with a label from $\{$***B, I, O***$\}$. Then the expertise of the k^{th} annotator can be modeled by a 3×3 matrix of parameters as shown below:

$$\begin{array}{c|ccc} & B & I & O \\ \hline B & \alpha_{k1} & \alpha_{k2} & 1-\alpha_{k1}-\alpha_{k2} \\ I & \beta_{k1} & \beta_{k2} & 1-\beta_{k1}-\beta_{k2} \\ O & \gamma_{k1} & \gamma_{k2} & 1-\gamma_{k1}-\gamma_{k2} \end{array} \quad (4)$$

where α_{k1} denotes the precision of the k^{th} annotator in annotating a token with correct label ***B***, α_{k2} denotes the probability that the answer is ***B*** but the annotator labels it with ***I***, accordingly, $1-\alpha_{k1}-\alpha_{k2}$ denotes the probability that the answer is ***B*** but the annotator labels it with ***O***. The parameters β and γ are defined with similar meanings as that of α. One can easily infer that the higher values the parameters on diagonal of Eq.(4) are, the more reliable is this annotator. Given K annotators, the total number of parameters of φ is $K \times 3 \times 3$.

With these parameters describing the annotator's reliability, we represent $\Pr(A_k^{(i)}|L_t^{(i)})$ in Eq.(5).

$$\Pr(A_k^{(i)}|L_t^{(i)}) = \prod_j \Pr(a_{ijk}|l_{ijt}) \quad (5)$$

where $\Pr(a_{ijk}|l_{ijt})$ can be represented by the parameters defined in the matrix of Eq.(4). For example, when l_{ijt} =***B*** and a_{ijk}=***I***, the probability $\Pr(a_{ijk}|l_{ijt})$ equals to α_{k2}.

Parameter Estimation

The problem now is to estimate the parameters ϑ and φ, and we achieve this objective by maximizing the likelihood in Eq. (6).

$$\begin{aligned} l &= \log \Pr(\mathcal{A}|\mathcal{X}) \\ &= \sum_{i=1}^{N} \log\left(\sum_t \prod_{k=1}^{K} \Pr(A_k^{(i)}|L_t^{(i)})\Pr(L_t^{(i)}|X^{(i)})\right) \quad (6)\end{aligned}$$

Since Eq. (6) involves the log calculation over the sum of products, it is intractable to estimate the parameters directly. To solve this problem, we employ EM (Expectation Maximization) method and introduce new hidden variables μ in Eq.(7).

$$\mu_{it} = \Pr(L_t^{(i)}|\{A_k^{(i)}\}, X^{(i)}) \quad (7)$$

After introducing the hidden variables, the log-likelihood could be transformed to Eq. (8)

$$l' = l'_1 + l'_2 \quad (8)$$

where

$$l'_1 = \sum_{i=1}^{N} \sum_{t} \mu_{it} \sum_{k=1}^{K} \log \Pr(A_k^{(i)} | L_t^{(i)}) \tag{9}$$

$$l'_2 = \sum_{i=1}^{N} \sum_{t} \mu_{it} \log \Pr(L_t^{(i)} | X^{(i)}) \tag{10}$$

In the E-step: μ is estimated as follows in Eq. (11).

$$\mu_{it} = \frac{\prod_{k=1}^{K} \Pr(A_k^{(i)} | L_t^{(i)}) \Pr(L_t^{(i)} | X^{(i)})}{\sum_t \prod_{k=1}^{K} \Pr(A_k^{(i)} | L_t^{(i)}) \Pr(L_t^{(i)} | X^{(i)})} \tag{11}$$

Since the parameter φ only appears in l'_1 and the parameter ϑ only appears in l'_2. In the M-Step, with the parameters u fixed, we maximize the loglikelihood l'_1 and l'_2 in Eq. (9) and Eq. (10) separately to calculate φ and ϑ.

For the parameters $\varphi = (\alpha, \beta, \gamma)$, it can be obtained easily by differentiating the log-likelihood in Eq. (9). For example,

$$\alpha_{k1} = \frac{\sum_{i=1}^{N} \sum_t \mu_{it} n_{it}^{k1}}{\sum_{i=1}^{N} \sum_t \mu_{it} (n_{it}^{k1} + n_{it}^{k2} + n_{it}^{k3})} \tag{12}$$

where n_{it}^{k1} denotes the number of tokens in $X^{(i)}$ with the label ***B*** in $L_t^{(i)}$ and the k^{th} annotator also labels it with ***B***. n_{it}^{k2} and n_{it}^{k3} are defined similarly. Thus the sum $n_{it}^{k1} + n_{it}^{k2} + n_{it}^{k3}$ denotes the number of the tokens that are labeled with ***B*** in $L_t^{(i)}$.

For the parameters ϑ, if we combine Eq. (2) and (10) together, it is easy to find that the maximization of Eq. (10) is indeed a weighted CRF parameter estimation problem. The input training set is in the form below:

$$\begin{array}{cccc} (X^{(1)}, L_1^{(1)}) & (X^{(1)}, L_2^{(1)}) & \ldots & (X^{(1)}, L_T^{(1)}) \\ (X^{(2)}, L_1^{(1)}) & (X^{(2)}, L_2^{(1)}) & \ldots & (X^{(1)}, L_T^{(2)}) \\ \cdots & \cdots & \cdots & \cdots \\ (X^{(N)}, L_1^{(N)}) & (X^{(N)}, L_2^{(N)}) & \ldots & (X^{(N)}, L_T^{(N)}) \end{array}$$

where each instance $(X^{(i)}, L_t^{(i)})$ is associated with a weight μ_{it}, therefore the log-likelihood of CRF optimization is modified to

$$l'_2 = \sum_{i=1}^{N} \Big(\sum_t \sum_{j,p} \mu_{it} \lambda_p f_p(l_{i(j-1)t}, l_{ijt}, x_{ij}) - \log(Z(X^{(i)})) \Big) \tag{13}$$

where $Z(X^{(i)}))$ can be calculated via Eq.(3).

The Eq. (13) can be calculated by the iterative scaling algorithms (IIS) proposed by (Lafferty, McCallum, and Pereira 2001). IIS works in an iterative manner and consecutively update the weights as $\lambda_p \leftarrow \lambda_p + \delta\lambda_p$. The complete process of parameter estimation in Sembler is listed in Algorithm 1.

Algorithm 1 first enumerates all the possible labeling on each instance $X^{(i)}$ and then adds every instance label pair

Algorithm 1: Parameter Estimation Process of Sembler

```
1  Set the training data set $D = \emptyset$.
2  foreach Instance $X^{(i)}$ do
3      Enumerate all possible labeling $\{L_1^{(i)}, L_2^{(i)}, \ldots\}$.
4      Add all instance label pairs $(X^{(i)}, L_t^{(i)})$ to $D$.
5  while EM not converged do
6      E step:
7          Update the values of hidden variable $\mu$ according
           to Eq. (11).
8      M step:
9          Update $\varphi$ according to Eq. (12).
10         Assign weights $\mu_{it}$ to each item in $D$.
11         Estimate the values of $\lambda_p$ with IIS method.
12         while IIS not converged do
13             Calculate $\delta\lambda_p$.
14             $\lambda_p \leftarrow \lambda_p + \delta\lambda_p$
```

$(X^{(i)}, L_t^{(i)})$ into the training data set D. If the average number of different labeling per instance is $\bar{T}$, the total size of the training set is thus $N \times \bar{T}$. Then the EM iteration is performed to estimate the values of ϑ and φ. Please note that the M-Step of Algorithm 1 includes parameter estimation of weighted CRF over the training data set D which also employs an iterative process. Therefore Algorithm 1 can be viewed as a level 2 nested iteration. If we represent the computational complexity of CRF parameter estimation with o(CRF), the dominant computational complexity of Sembler is $I \times o$(CRF) where I denotes the number of EM iterations. Since Sembler is performed off line and the parameter estimation process of CRF can be accelerated and parallelized (Vishwanathan et al. 2006), the computational complexity is not a barrier for practical use of Sembler.

Generation of All Possible Sequential Labelings

In the formulation above, for each instance $X^{(i)}$, we need to generate all possible sequential labelings $\{L_1^{(i)}, L_2^{(i)}, \ldots\}$. A straight-forward approach is to enumerate all label combinations. However such an approach will cause combinatorial explosion and lead to difficulties in computation. For example, if the label Alphabet is {***B, I, O***}, the total number of sequential labeling is as large as 3^M in a sentence with M tokens. Besides, many sequential labelings generated by brute-force enumeration are not valid. For example, in NER tasks, the label sequence (***O, I***) is incorrect and label a token that is obviously not part of a named entity with ***B*** or ***I*** is inappropriate. To reduce the size of sequential labelings and avoid unnecessary compututational cost, we propose Algorithm 2 to generate valid sequential labelings.

To demonstrate how the above algorithm works, we examine the following example:

Sentence 1:	activation	induced	by	FGF1
Annotator 1:	***O***	***O***	***O***	***B***
Annotator 2:	***O***	***B***	***O***	***O***

Sentence 1 is annotated by two annotators with different la-

Algorithm 2: Generating Valid Sequential Labelings

```
1 foreach $A_k^{(i)}$ on $X^{(i)}$ do
2     foreach token $w_j$ in $X^{(i)}$ do
3         Restore the label $a_{ijk}$of $A_k^{(i)}$ on $w_j$.
4 foreach token $w_j$ in $X^{(i)}$ do
5     Reserve the labels on $w_j$ the frequency of which is
      above a preset threshold.
6     Remove other labels.
7 Generating sequential labeling with the remaining
  labels on each token.
```

belings. If we set the threshold to be 1, four possible sequential labelings will be generated as follows:

Sentence 1:	activation	induced	by	FGF1
Labeling 1:	***O***	***O***	***O***	***O***
Labeling 2:	***O***	***O***	***O***	***B***
Labeling 3:	***O***	***B***	***O***	***O***
Labeling 4:	***O***	***B***	***O***	***B***

Experiments

We evaluate the proposed approach from two main perspectives: (1) Does it outperform the baseline ensemble methods, such as majority voting; (2) Does its performance remain stable in different settings of annotator size and individual expertise. We evaluate the first perspective on a real Twitter NER data set, and generate synthetic data sets with varied expertise settings to explore the second.

Twitter Data Set

We use the Tweet dataset provided by (Finin et al. 2010). They publish 441 tweets on Amazon Mechanical Turk and invite annotators to label the appearance of organization, address and person names. In total, they received 4,791 annotations from 269 different contributors. That is 10 annotations for each tweet. Finin et al. (2010) also provides the ”Gold” labeling answers for these 441 tweets. The number of organizations, persons, and places in the tweet data set are 170, 171 and 167 respectively.

For each type of named entities, we apply the following three methods to the Tweet data set separately. Table 2 lists the F1 value of the annotations given by these three methods on the golden standard.

- Random Select (RS): For each tweet, we randomly select one of its annotators with equal probability and use his annotation as the result.
- Majority Vote (MV): For each word in tweet, we select the label that the majority of its annotators agree on.
- Sembler: the proposed method. For each word in a tweet, we select the common features used in typical named entity recognition tasks, including the adjacent words, the prefixes and suffixes, capitalization, etc. We treat these features as our observed knowledge over input word sequence and include them in Eq. (2). After preforming parameter estimation step, we apply the acquired parameter ϑ to relabel the input tweets which follows the CRF inference process.

From Table 2, we observe that the proposed *Sembler* method outperforms the rest two for all named entities. Most importantly, the advantage in performance is more significant when the quality of crowd annotations is poor. For example, in the “Organization” named entity, the majority of the annotators provide wrong labels. But in case of random selection, its possible that the correct label from the minorities is chosen. As a result, the *Majority Vote* method performs even worse than *Random Select*. In this condition, *Sembler* improves the f1 values by more than 50% compared to *Majority Vote*. This is because when majority agreement does not work, the *Sembler* method instead relies on labeled organization names to recognize other names with similar features. Therefore Sembler has both higher precision than majority vote and the higher recall than random select at the same time.

BioCreative Data Set

To explore the performance of the proposed method in various circumstances, we need to simulate crowd annotations with different settings on the number of annotators and expertise. We generate a synthetic data set from BioCreative (Hirschman et al. 2005), which consists of 7,500 Medline abstracts. In each abstract, the appearances of gene names are labeled by domain experts. Therefore, this data set can be viewed as a golden standard for gene name identification. We randomly select 400 sentences from BioCreative to generate synthetic crowd annotations. From our knowledge on gene name annotation, three types of mistakes are easily made by annotators.

- Ignore Error: due to the pressure of manual annotating, some gene name appearances could be missed by annotators.
- Boundary Error: due to the complexity of gene names, some inexperienced annotators could not label the starting position and ending position of gene names correctly. Four types of mistakes are common: the inclusion of an extra token that lies before/after the gene name; the exclusion of the first/last token of a gene name.
- Split Error: since some gene names occupy several tokens, a single gene name can be mistaken for several adjacent gene names.

To include the above three types of common mistakes, we generate synthetic crowd sequential labeling in BioCreative according to the following steps:

1. For each annotator a, we assign a probability p to represent his annotating expertise.
2. For each appearance of a gene name in the sentence, this annotator has p probability to correctly label this gene name and $1 - p$ probability to make mistakes.
3. When this annotator is chosen to make mistakes on a gene name, the above three types of errors are picked with

Canada	Organization			Person			Address		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
RS	0.402	0.183	0.252	0.524	0.532	0.528	0.658	0.477	0.553
MV	**0.667**	0.118	0.200	0.610	0.631	0.621	**0.823**	0.557	0.664
Sembler	0.660	**0.215**	**0.324**	**0.612**	**0.714**	**0.659**	0.800	**0.599**	**0.685**

Table 2: Comparison Results between Three Methods on Twitter NER Corpus

equal probability. For example, the *Ignore Error* can happen with probability $\frac{1-p}{3}$. When it occurs, the labels on each token of this gene are changed to $\boldsymbol{O}$.

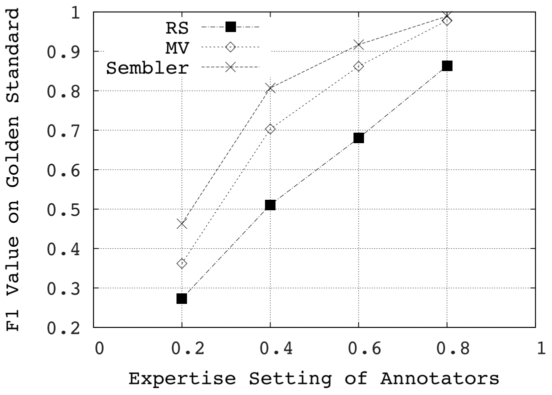

Figure 1: Comparative Evaluation on Synthetic BioCreative Datasets

Figure 1 summarizes the comparison results among three methods on the BioCreative data set where the number of annotators is set to be 5. We observe that with different setting of annotator expertise, the proposed *Sembler* method consistently outperforms the other two baselines. Similar to previous experiment on Twitter data set, the performance advantages become much more significant when the quality of crowd annotating is poor, i.e., with smaller p.

We also explore whether the number of annotators can affect the performance of the proposed method. We set the number of annotators to 6,8 and 10 respectively, and generate synthetic data sets with different expertise settings. Table 3 lists the comparison results of three methods. From this table, we find that under different annotator number settings, *Sembler* consistently outperforms the other two baselines. The number of annotators can help to improve the accuracy when the expertise of each annotator is reliable which is in compliance with the conclusion of (Sheng, Provost, and Ipeirotis 2008).

Related Works

There have been increasing interests to collect manual annotations for NLP tasks via crowdsourcing services. In (Snow et al. 2008), the annotators from MTurk are invited to work on five NLP annotating tasks, including affect recognition, word similarity, recognizing textual entailment, event temporal ordering, and word sense disambiguation. They find that a small number of repeated non-expert annotations could match the performance of an expert annotator. In (Su et al. 2007), the annotators provided labels for hotel name entity entities as well as other information extraction tasks. They also found that the crowd annotations are of high accuracy. In (Yetisgen-Yildiz et al. 2010), the annotators are invited to label medical names which requires specific domain knowledge. Surprisingly, the quality of acquired crowd annotations are still satisfactory. In (Callison-Burch 2009) and (Evanini, Higgins, and Zechner 2010), they prove the usefulness of crowdsourcing in complex NLP tasks, such as document translation and speech transcription.

Annotator	Method	$p=0.2$	$p=0.4$	$p=0.6$
K=6	RS	0.312	0.504	0.683
	MV	0.477	0.748	0.948
	Sembler	**0.582**	**0.806**	**0.962**
K=8	RS	0.273	0.475	0.668
	MV	0.424	0.797	0.947
	Sembler	**0.487**	**0.849**	**0.968**
K=10	RS	0.279	0.499	0.697
	MV	0.450	0.750	0.968
	Sembler	**0.557**	**0.831**	**0.983**

Table 3: F1 Value of Three Methods on BioCreative Data Set with Different Settings of Annotator Expertise

Although the crowd annotations have been proven useful for many NLP tasks. Due to the unknown expertise and devotion of annotators, some errors caused by unintentional or deliberate reasons are inevitable. Therefore, many approaches have been proposed to remove these errors while preserving correct annotations. In (Zaidan and Callison-Burch 2011), they provide several linguistic and statistical measures to discriminate between acceptable and unacceptable crowd translations; In (Sheng, Provost, and Ipeirotis 2008), they explore the use of noisy labels in training classifiers. They provide answers to two questions, (1) how to choose which example should get more labels; (2) how to consider the label's uncertainty; In (Dempster, Laird, and Rubin 1977), they propose a maximum likelihood based statistical framework to model the labels from multiple sources and infer the correct label for each item; (Raykar et al. 2009) further extends the framework proposed by (Dempster, Laird, and Rubin 1977) and integrate it with the classification process. As a result, a classifier can be trained directly from crowd annotations.

In previously proposed approaches, the crowd annotations are abstracted as either a numerical rating or a binary selection problem, each annotation is assumed independent from each other. Therefore the dependency information are ignored in refining crowd annotations. Different from previous approach, we propose an ensemble method that is able to model the dependencies among annotations, as well as other factors, like majority vote and individual annotator's expertise.

Conclusion

In this paper, we present a statistical model *Sembler* to ensemble crowd sequential labeling. With help of this model, we can either acquire a refined and unified sequential labeling over the input data set, or train a sequential data annotator directly from crowd annotations. Therefore the proposed framework can benefit many sequential labeling tasks including named entity recognition (NER), part of speech (POS) tagging and word segmentation.

Similar to previous refinement approaches that are targeted for numeric rating or binary selection, the proposed framework model the expertise of individual annotators and follow the majority vote principal as well. Importantly, the proposed framework also considers the statistical dependencies and co-relation between sequential data. As a result, the labels on all entities in an instance are considered as a whole in the ensemble process. Besides, correct annotation in one instance enhances the correctness of other annotations which share similar linguistic or contextual features. We evaluate the proposed model on a real Twitter and a synthetical biological data set, and find that the proposed method outperform baseline methods, such as majority vote, consistently in various environment settings. Interestingly, the proposed *Sembler* works particuarly well when the quality of crowd annotation are poor.

References

Callison-Burch, C. 2009. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, 286–295. Stroudsburg, PA, USA: Association for Computational Linguistics.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 39(1):1–38.

Evanini, K.; Higgins, D.; and Zechner, K. 2010. Using amazon mechanical turk for transcription of non-native speech. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, 53–56. Stroudsburg, PA, USA: Association for Computational Linguistics.

Finin, T.; Murnane, W.; Karandikar, A.; Keller, N.; Martineau, J.; and Dredze, M. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, 80–88. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hirschman, L.; Yeh, A.; Blaschke, C.; and Valencia, A. 2005. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC bioinformatics* 6 Suppl 1.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 282–289. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

McCallum, A.; Freitag, D.; and Pereira, F. C. N. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, 591–598. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rabiner, L. R. 1990. Readings in speech recognition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. chapter A tutorial on hidden Markov models and selected applications in speech recognition, 267–296.

Raykar, V. C.; Yu, S.; Zhao, L. H.; Jerebko, A.; Florin, C.; Valadez, G. H.; Bogoni, L.; and Moy, L. 2009. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 889–896. New York, NY, USA: ACM.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, 614–622. New York, NY, USA: ACM.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. Y. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, 254–263. Stroudsburg, PA, USA: Association for Computational Linguistics.

Su, Q.; Pavlov, D.; Chow, J.-H.; and Baker, W. C. 2007. Internet-scale collection of human-reviewed data. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, 231–240. New York, NY, USA: ACM.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Schmidt, M. W.; and Murphy, K. P. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, 969–976. New York, NY, USA: ACM.

Yetisgen-Yildiz, M.; Solti, I.; Xia, F.; and Halgrim, S. R. 2010. Preliminary experience with amazon's mechanical turk for annotating medical named entities. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, 180–183. Stroudsburg, PA, USA: Association for Computational Linguistics.

Zaidan, O. F., and Callison-Burch, C. 2011. Crowdsourcing translation: professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 1220–1229. Stroudsburg, PA, USA: Association for Computational Linguistics.