Active Learning with Partially Annotated Sequence
(部分的アノテーションを用いた能動学習)

Dittaya Wanvarie

Department of Computational Intelligence and Systems Science

Tokyo Institute of Technology

A thesis submitted for the degree of

Doctor of Philosophy (Ph.D.)

September 2011

Abstract

This thesis presents an active learning framework for sequence labeling task with the confidence re-estimation. The framework can reduce the annotation cost from the whole corpus to a partial corpus, while achieving the classifier with similar accuracy. Active learning consists of 3 main parts, the sampling, the annotation, and the re-training phases. The confidence re-estimation can be augmented in all phases. The re-estimation in active learning reduces the annotation cost in terms of labeled tokens. The simplified sampling is also proposed to reduce the annotation cost in terms of computational time.

Acknowledgements

First and foremost, I owe my deepest gratitude to Prof. Manabu Okumura, who gave me the first opportunity to pursue my study in this laboratory. From the start, he consistently give me valuable advices and total supports on my research.

I am also indebted to Assoc. Prof. Hiroya Takamura, who always have brilliant ideas and excellent suggestions. It is an honor to me to have him as a co-advisor. I would like to thank Dr. Kritsada Sriphaew for his encouragement, discussions and assistance in my work. This thesis could not have been accomplished without help from my colleagues at Okumura laboratory, especially the group of system administrators.

During years of study, I receive the financial support from the Thai government through the Strategic Frontier Research Scholarship. Without this support, my opportunity to pursue this study might be difficult. Finally, I wish to thank my family, friends, and the Thai community in Japan who always give me moral support through these years of study.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1

Introduction

A sequence is an important structure in the real world. A gene in Figure 1.1[1] is an example of a sequence structure in the biological domain. A gene is a subsequence of DNA or RNA in a chromosome, which represents a genetic trait of a living organism. There are also relationships among some diseases, genes, and the medicine [28]. The study of genes can lead to the invention of a new medicine to cure a disease.

A sentence in a human language is in the form of a word sequence. The research on human languages is called Natural Language Processing (NLP), which is the main focus of this thesis. We can extract much information from a sequence of words such as the intention of the speaker [25], the person name noted in a text document [19].

Figure 1.2 is an excerpt from the Naoto Kan page in Wikipedia[2]. We try to find all person names and organization names in the text. Since a name can be either a single word or a sequence of words, finding names is equivalent to labeling a word or a word sequence which are names.

More specifically, assigning a label to each unit in a sequence is called *annotation*. However, annotating all data by hand is not practical. Although a small text passage may contain hundreds of words, a newspaper, for example, contains several passages and requires a large amount of human effort to scan through the data collection. The number of human genes is not exactly known yet but the recent estimation is approximately twenty thousand genes [12]. If an expert need to annotate all data every time the new data is presented, the annotation cost will become intractable.

[1] http://upload.wikimedia.org/wikipedia/commons/0/07/Gene.png Retrieved on June 21, 2011.
[2] http://en.wikipedia.org/wiki/Naoto_Kan#Career Retrieved on June 21, 2011.

1. INTRODUCTION

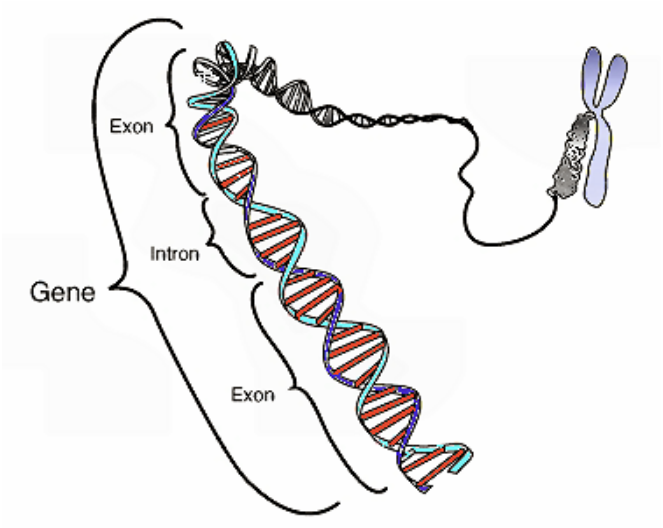


Figure 1.1: Gene data

Career [edit]

After graduating from university, Kan worked at a patent office for four years.[3] He actively engaged in civic grassroots movements for years and also served on election campaign staff for Fusae Ichikawa, a women's rights activist.[4]

After having lost in 1976, 1979 general elections and 1977 Upper House election, Kan finally achieved a seat in the lower house in 1980 as a member of Socialist Democratic Federation. He gained national wide popularity in 1996, when serving as the Minister of Health and Welfare, admitting government's responsibility for the spread of HIV-tainted blood in 1980s and directly apologized to victims. At that time, he was a member of a small party forming the ruling coalition with the Liberal Democratic Party (LDP). His frank action was completely unprecedented and was applauded by the media and the public.[*citation needed*]

In 1998, his image was affected by allegations of an affair, vigorously denied by both parties, with a television newscaster and media consultant, Yūko Tonomoto.[5] After Yukio Hatoyama resigned as the leader of the Democratic Party of Japan (DPJ), Kan again took over the position. In July 2003, the DPJ and the Liberal Party led by Ichirō Ozawa agreed to form a uniformed opposition party to prepare for the general election that was anticipated to take place in the fall.

During the campaign of the election of 2003, the DPJ called the election as the choice of the government between the ruling LDP-bloc and the DPJ, with Kan being presented as the alternative candidate to then Prime Minister Junichiro Koizumi. His face was used as the trademark of the campaign against the LDP.[*citation needed*]

However, in 2004 Kan was accused of unpaid annuities and forced to again resign the position of leader. On 10 May 2004 he officially announced his resignation and made the Shikoku Pilgrimage. Later, the Ministry of Health, Labour and Welfare spokesman apologized, saying the unpaid record was due to an administrative error.

In mid-October 2005, Kan, who turned 60 in 2006, proposed the creation of a new political party to be called the "Dankai (baby boomer) Party." The initial intent of the party was to offer places of activity for the Japanese baby boomers - 2.7 million of whom began to retire *en masse* in 2007.

He believes the Japan Self-Defense Forces should play a more prominent role on the international stage.[6]

Figure 1.2: Text data

Recently, we can train a machine expert to perform this annotation task instead of a human expert. A machine which can predict the output of the given input is called a classifier. Various branches of training algorithms are well studied in machine learning fields. Generally, the training algorithms are divided into the following three broad classes:

- Supervised learning. A human gives the supervision to the machine by annotating the correct output of the given data. There are adjustable parameters in each learning algorithm. We try to adjust these parameters through the training to make the machine correctly predict the output.

- Unsupervised learning. A machine can learn to predict the correct output from a raw data collection without any human supervision. Learning algorithms in a class will induce the underlying information or structure from the observed input data. For example, a clustering algorithm will group a set of instances which are similar to each other.

- Semi-supervised learning. A human gives some supervision to a part of the training collection. A machine can learn from both the annotated and unannotated data.

This thesis is concerned with the semi-supervised learning approach. Since the size of data collection is still extremely large, human supervision is still costly even when we have to annotate the entire data collection only once to train the machine. For example, the Penn treebank contains almost 5 million English words. The part-of-speech annotation requires approximately 22 minutes per 1000 word annotation [32]. If there is a single expert, annotating the entire treebank requires 180,000 hours. The difficulty of the annotation also depends on the task and the data set. The speed of the syntactic tree annotation in the Penn treebank is approximately 400 words per hour. Since syntactic annotation is more complex than the part-of-speech annotation, the time for the syntactic annotation is clearly longer than the time for the part-of-speech annotation.

Moreover, the accuracy of the machine also depends on the distribution of training instances in the data set. A classifier trained on a data set but tested on different data set usually achieves low-accuracy [34]. Even though the text collections are from

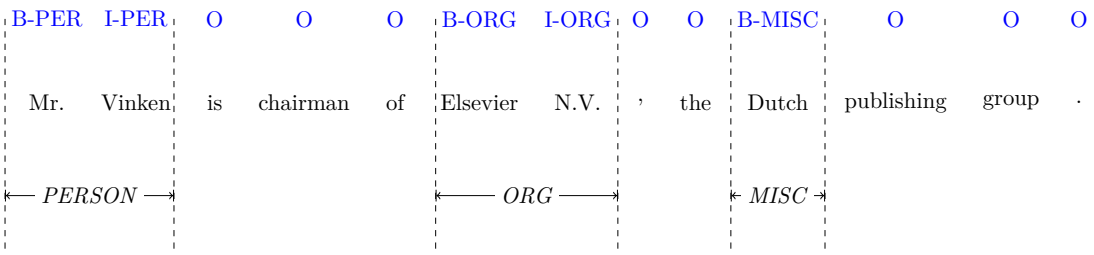


Figure 1.3: Named entity recognition

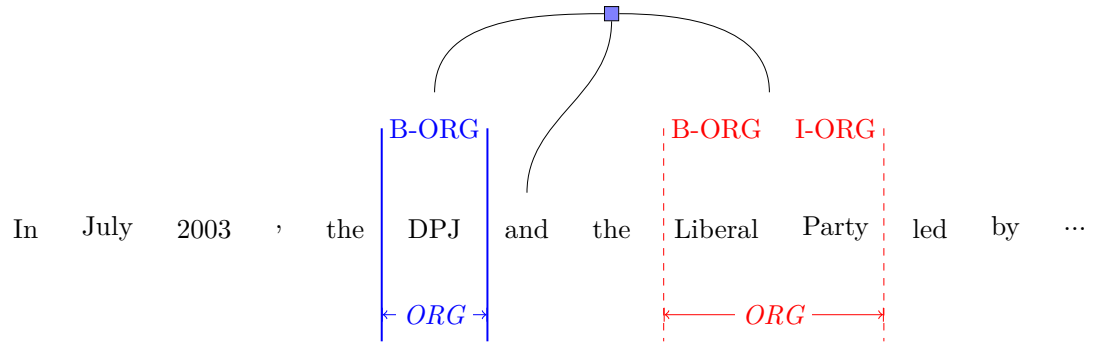


Figure 1.4: Dependencies among output labels

the same languages, there are other differences such as writing styles or document topics, which will affect the accuracy of the classifier. The number of new data sets are countless. When a new data set is provided, we cannot avoid manual annotation of new data, but we will try to minimize the annotation cost from the entire data set to a part of data set.

We formulate the labeling task as a sequence labeling. The objective of sequence labeling is similar to the other classification tasks, which is to find the output for the given input. A classifier is trained to predict the desired output of the given input. Figure 1.3 shows an example of the sequence labeling task in NLP, the named entity recognition task. There are 3 types of named entities in this figure, *PERSON*, *ORG*, and *MISC*. Since some entities consist of several words, we introduce 2 label types for each entity. the beginning word in the entity (*B*-chunk), and other words in the entity (*I*-chunk).

However, the sequence labeling task has a structured output which is constructed from several substructures with some dependencies among them. Figure 1.4 shows a dependency between the output of *DPJ* and *Liberal Party*. The semantic rule of the

word *and* between these two words indicates that both words should have the same output label. The structured output increases the number of possible output to be extremely large, in a very sparse output space. This sparse output space will decrease the accuracy of the classifier. One solution to this problem is to factorize the structure and perform the classification on its substructure. However, the dependencies among the substructures make the factorization not be trivial. Although we can assume that there is no dependency among substructures, we will show in the experiment that neglecting these dependencies leads to poor prediction accuracy.

The dependencies among the output labels of a sequence make the annotation more difficult than the annotation of a non-structured instance. The following example is the part-of-speech labeling task. Let us have a sentence "*There is a trap.*" which contains 4 words:

1. *There*, which is either an adverb (here and there) or a pronoun.
2. *is*, which is a verb.
3. *a*, which is a determiner.
4. *trap*, which is either a noun or a verb.

We will find part-of-speech label of each word. If a word *trap* stands alone, it can be either a noun or a verb. However, the part-of-speech of the word *trap* in this sentence depends on the part-of-speech of its previous words, according to the syntactic rules. If we consider the whole structure as a single input, there are 4 possible outputs;

1. ADV V DET N.
2. ADV V DET V.
3. PRON V DET N.
4. PRON V DET V.

The number of possible outputs will substantially increase when the sentence is long. However, the frequency of each output is extremely low. In other words, the output space is extremely sparse. This sparseness of output labels in the training set will decrease the accuracy of the classifier trained on this training set. We can reduce the

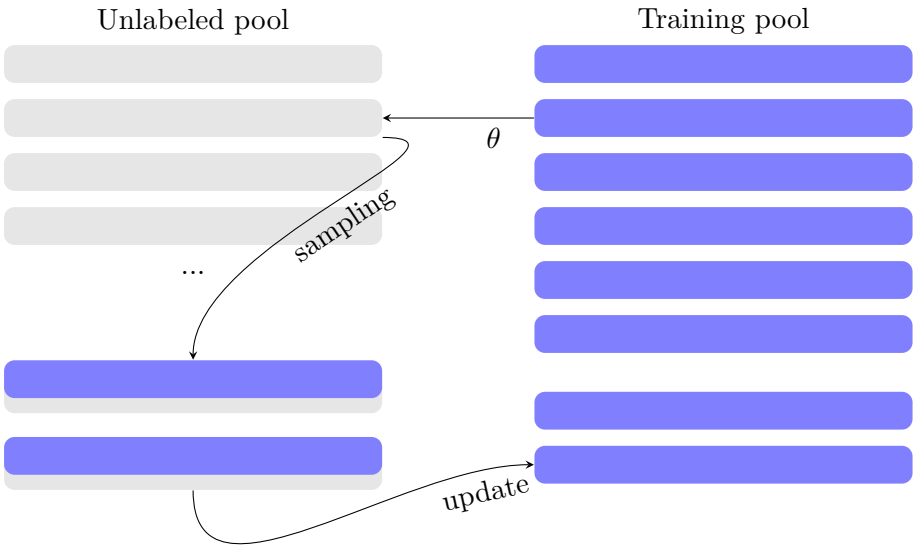


Figure 1.5: A general pool-based active learning framework

sparseness of the output by factorizing the structure into substructures, and perform the prediction in substructure level, but keep the dependency among the substructures in the consideration.

Active learning is a learning framework which can reduce the annotation cost from the entire training set to a part of the training set [14]. The hypothesis behind active learning is that the informativeness of each instance in the training set is not equal. The informativeness of the instance is the contribution to the accuracy of the classifier if the instance is annotated, and added to the training set. Active learning is an iterative framework which gradually selects informative instances from the entire data set and adds to the training pool as shown in Figure 1.5. In order to reduce the annotation cost, we need to select only the highly informative instances. Therefore, we can reduce the annotation cost but keep the high level of accuracy. The active learning framework is also applicable to the sequence labeling task. The informative instance can either be the whole sequence [44] or a part of sequence [53]. This thesis adopted the prediction confidence of a token as the informativeness score. When the prediction confidence of a token is low, it indicates that the token may contain crucial information which is not previously known to the tagger. Hence, we define such a token to be informative.

The dependency among output substructures also affects the annotation. When a token in a sequence is annotated, this label information will affect the prediction confidence of its neighboring tokens. In order to reduce the annotation cost of a sequence, we can incorporate this information propagation idea into the annotation task. We call the idea *Confidence re-estimation*. A general active learning framework is initially proposed to a simple classification task whose output is not a structure. Therefore, we can incorporate the confidence re-estimation into the annotation of the substructure, the training of a classifier, also the sampling strategy.

This thesis proposes to incorporate the re-estimation of confidence in every step of an active learning framework. Furthermore, we analyze several parameters of active learning, which will affect the annotation cost. We also propose the efficient sampling strategy, which can reduce the computational cost from the re-estimation of confidence.

The outline of this thesis begins with the machine learning method adopted in this thesis, the conditional random fields (CRFs), followed by the description of a general active learning framework. Chapter 4 describes the research work related to the proposed framework in this thesis. Chapter 5 introduces the implementation of confidence re-estimation idea in each step of the active learning framework, and the efficient sampling strategy. Chapter 6 presents the experiments and analysis of the results. Finally, we conclude the contribution of this thesis and suggest the future work in the last chapter.

Chapter 2

Conditional Random Fields

The objective of a supervised classification task is to train a classifier to predict the human supervised output from the given input. In a probabilistic model, we can represent the input and output with random variables. Let x and y represent the input and output, respectively. We can draw a graph representation of the relationship between the input and output. Each vertex represents a random variable, while an edge indicates that there is a relationship between vertices.

Because of the sparseness of the structured output, we reformulate the structured output prediction to the substructure prediction. However, we still need to maintain the dependency among the substructures. A graphical model naturally handles this problem by modeling the dependency with the vertex and edge representation. Therefore, we adopt the graphical model in this thesis.

2.1 Graphical Model

A graphical model is a probabilistic model for which a graph denotes the conditional independence structure between random variables [7]. This thesis employs a particular

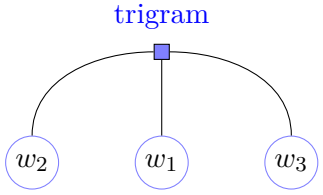
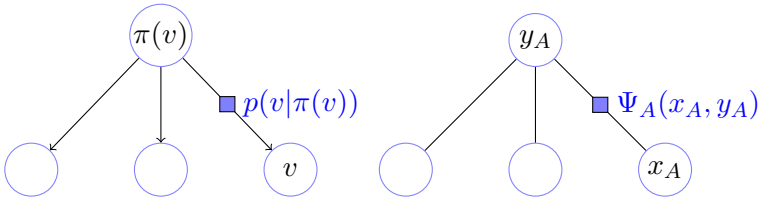


Figure 2.1: A trigram relationship



(a) Directed model **(b)** Undirected model

Figure 2.2: Graphical model: A factor graph

type of the graphical model, a *factor graph*. A factor graph is a bipartite graph. There are 2 vertex types in a factor graph, the variable vertices and the factor vertices. The variable vertices are input and output vertices, while the factor vertices are the relationships or dependencies among the variables.

We factorize the entire graph G into a number of factors. Each factor consists of a sub graph A whose variable vertices are bound together with one relationship. For example, a trigram relationship in Figure 2.1 requires 3 word vertices. A trigram relationship is a factor vertex, while 3 word vertices are variable vertices.

In a factor graph, the score of each factor is independent of the other factors. We can calculate the probability of G by the product of factors:

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{A} \Psi_A(\mathbf{x}_A, \mathbf{y}_A) \ , \tag{2.1}$$

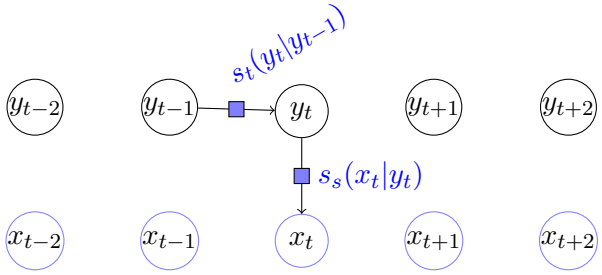
where Ψ is a factor function spanned on a sub graph A. The constant Z is the normalization function:

$$Z = \sum_{\mathbf{x},\mathbf{y}} \prod_{A} \Psi_A(\mathbf{x}_A, \mathbf{y}_A) \ . \tag{2.2}$$

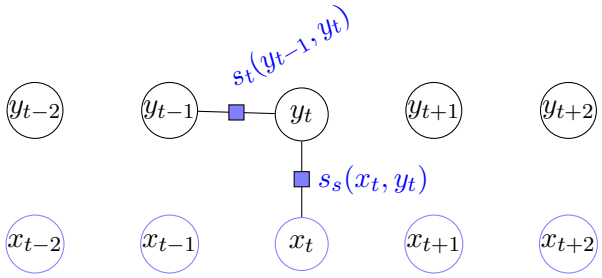
Figure 2.2 shows 2 types of a factor graph, the directed model and the undirected model. Variables vertices are in circle shapes while factor vertices are in square shapes. The difference between these models is the dependence assumption behind the factor score. A directed model is also known as a Bayesian network [22], which is based on the joint probability, $p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$. A directed model is factorized to be

$$p(\mathbf{x}, \mathbf{y}) = \prod_{v \in V} p(v|\pi(v)) \ , \tag{2.3}$$

where $\pi(v)$ is the parent of a vertex v in the graph G.



(a) Directed model



(b) Undirected model

Figure 2.3: Sequence model

On the other hand, an undirected model is based on the conditional probability, $p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})$. An undirected model is factorized to be

$$p(\mathbf{x}, \mathbf{y}) = \prod_{v \in V} p(v, \pi(v)) \ . \tag{2.4}$$

We can adopt the graphical model on a classification task. For a single output classification, the directed version of the probabilistic model can be described in the form

$$p(\mathbf{x}, y) = p(y) \prod_{A} p(x_A|y), \tag{2.5}$$

which is a naive Bayes classifier [20]. An example of the undirected graphical version is a logistic regression classifier, which is in the following form:

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\sum_A \phi_A(\mathbf{x}_A, y)} \ . \tag{2.6}$$

2.2 Linear Chain Conditional Random Fields

A basic classification only has a single output y. In a structured output problem, the output is a set of labels with some dependencies among them. For a sequence

labeling problem such as in Figure 2.3, the input and output are in the sequence form. Each output label corresponds to an input token. One choice of the dependency modeling for a sequence labeling problem is the Markov assumption, which assumes that the output y_t of a token x_t depends on its limited numbers of predecessor outputs $(y_{t-i}; i \in \{1, .., t-1\})$, and is independent of all other early outputs. For example, the 1^{st} order Markov assumption assumes that the probability of a sequence $\mathbf{y}$ is

$$p(\mathbf{y}) = \prod_{t=2}^{T} p(y_t|y_{t-1}) \ , \tag{2.7}$$

where T is the length of a sequence. Figure 2.3 also shows the dependency modeling under the 1^{st} order Markov assumption.

Hidden Markov model (HMM) adopted the 1^{st} order Markov assumption and the assumption that the input token depends on only its output label. HMM factorizes the joint probability $p(\mathbf{x}, \mathbf{y})$ between the input sequence $\mathbf{x}$ and the output sequence $\mathbf{y}$ to

$$p(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{T} p(y_t|y_{t-1})p(x_t|y_t), \tag{2.8}$$

where T is the sequence length. t is a position in the sequence. Figure 2.3a also shows the factor graph of HMM in equation (2.8).

Similar to the naive Bayes and logistic regression classifier pairs, HMM is the directed counterpart of the linear chain conditional random fields (linear CRFs). The linear chain CRFs [27] model the conditional probability of output label sequence $\mathbf{y}$ given input sequence $\mathbf{x}$ in the form of

$$P_\theta(\mathbf{y}|\mathbf{x}) = \frac{e^{\theta \cdot \phi(\mathbf{x},\mathbf{y})}}{Z_{\theta,\mathbf{x},\mathbf{Y}}} \ , \tag{2.9}$$

where $(\mathbf{x}, \mathbf{y}) : \mathbf{X} \times \mathbf{Y} \to \mathbb{R}^d$ is a function from a pair of input sequence $\mathbf{x}$ and output sequence $\mathbf{y}$ to a feature vector of d dimensions. Graphically, ϕ is defined on an arbitrary graph A, which is a subgraph of G. $Z_{\theta,\mathbf{x},\mathbf{Y}}$ is the normalization factor:

$$Z_{\theta,\mathbf{x},\mathbf{Y}} = \sum_{\mathbf{y} \in \mathbf{Y}} e^{\theta \cdot \phi(\mathbf{x},\mathbf{y})} \ .$$

Let α_j be the score of the prefix sequence until position j, called the forward score:

$$\alpha_j(y') = \sum_{y''} \alpha_{j-1}(y'')s_t(y'', y')s_s(y').$$
$$\alpha_1(y') = s_s(y') \ .$$

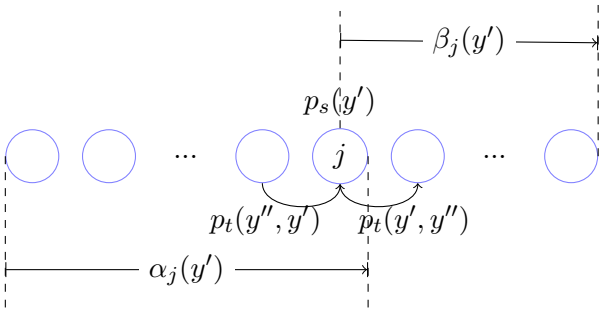


Figure 2.4: α and β calculation

Let β_j be the score of the suffix sequence from position j, called the backward score:

$$\beta_j(y') = \sum_{y''} s_t(y', y'') s_s(y'') \beta_{j+1}(y'').$$
$$\beta_T(y') = 1 \ .$$

α and β calculation are shown in Figure 2.4. $s_t(y', y'')$ is the transition score from label y' to label y''. $s_s(y')$ is the output score of label y'. We can efficiently compute $Z_{\theta,\mathbf{x},\mathbf{Y}}$ by

$$Z_{\theta,\mathbf{x},\mathbf{Y}} = \sum_{y' \in Y_1} \alpha_1(y') \cdot \beta_1(y') \ ,$$

where Y_1 is all possible labels of y_1.

$\theta \in \mathbb{R}^d$ is a vector of model parameters. For a set of N training sequences $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, the learning process will maximize the following log likelihood function:

$$LL(\theta) = \sum_{n=1}^{N} ln(P_\theta(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})) \ . \tag{2.10}$$

We can apply standard optimization techniques such as L-BFGS in [31] or SGD in [58], to the objective function in equation (2.10).

2.3 Conditional Random Fields for Partially Annotated Sequences

The maximization of the objective function in equation (2.10) requires all tokens in the input sequence to be labeled since some factors depend on the output labels. If a sequence is partially labeled, we will not be able to calculate the score of those factors.

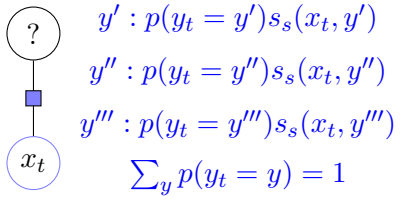
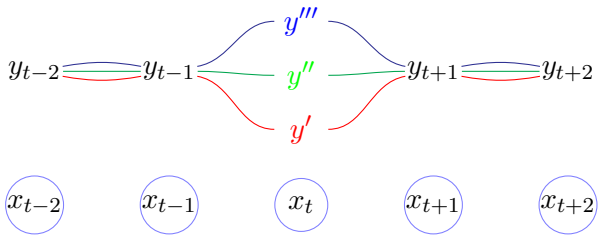


Figure 2.5: Marginalized label



$$\mathbf{L} = ((x_{t-2}, y_{t-2}), (x_{t-1}, y_{t-1}), (x_t), (x_t, x_{t+1}, y_{t+1}), (x_{t+2}, y_{t+2}))$$

Figure 2.6: A partially annotated sequence

One solution to this problem is to estimate the output label of all unlabeled tokens using the marginalized outputs [5; 56]. Figure 2.5 shows the marginalized labels of the token x_t. There are 3 possible labels, y', y'', and y'''. Each output has a marginalized probability $p(y')$, $p(y'')$, and $p(y''')$, respectively. We assign all possible output labels with the corresponding marginal probability to x_t.

Given a partially labeled sequence or ambiguously labeled sequence $\mathbf{L}$, let $\mathbf{Y_L}$ be the set of all possible output sequences consistent with $\mathbf{L}$. For example, Figure 2.6 consists of 5 input tokens, x_{t-2} to x_{t+2}. All tokens except x_t have a single output label. x_t has 3 possible output labels, y', y'', and y'''. Hence, $\mathbf{Y_L}$ consists of 3 sequence pairs:

- $(\mathbf{x}, (y_{t-2}, y_{t-1}, y', y_{t+1}, y_{t+2}))$
- $(\mathbf{x}, (y_{t-2}, y_{t-1}, y', y_{t+1}, y_{t+2}))$
- $(\mathbf{x}, (y_{t-2}, y_{t-1}, y', y_{t+1}, y_{t+2}))$.

We can estimate the probability of $\mathbf{Y_L}$ given $\mathbf{x}$ by

$$P_\theta(\mathbf{Y_L}|\mathbf{x}) = \sum_{\mathbf{y} \in \mathbf{Y_L}} P_\theta(\mathbf{y}|\mathbf{x}) \ . \tag{2.11}$$

Using equation (2.11), the log likelihood in equation (2.10) is modified to

$$\begin{aligned} LL(\theta) &= \sum_{n=1}^{N} \ln P_\theta(\mathbf{Y}_{\mathbf{L}^{(n)}} | \mathbf{x}^{(n)}) \\ &= \sum_{n=1}^{N} (\ln \sum_{\mathbf{y} \in \mathbf{Y}_{\mathbf{L}^{(n)}}} \frac{e^{\theta \cdot \phi(\mathbf{x^{(n)}}, \mathbf{y})}}{\sum_{\mathbf{y}' \in \mathbf{Y}} e^{\theta \cdot \phi(\mathbf{x}^{(n)}, \mathbf{y}')}}) \\ &= \sum_{n=1}^{N} (\ln \sum_{\mathbf{y} \in \mathbf{Y}_{\mathbf{L}^{(n)}}} e^{\theta \cdot \phi(\mathbf{x^{(n)}}, \mathbf{y})} - \ln \sum_{\mathbf{y}' \in \mathbf{Y}} e^{\theta \cdot \phi(\mathbf{x}^{(n)}, \mathbf{y}')}) \\ &= \sum_{n=1}^{N} (\ln Z_{\theta, \mathbf{x}^{(n)}, \mathbf{Y}_{\mathbf{L}^{(n)}}} - \ln Z_{\theta, \mathbf{x}^{(n)}, \mathbf{Y}}) \ . \end{aligned} \tag{2.12}$$

$\mathbf{x}^{(n)}$ and $\mathbf{L}^{(n)}$ are n^{th} input sequence and a set of all possible output sequences, respectively. $Z_{\theta, \mathbf{x}^{(n)}, \mathbf{Y}_{\mathbf{L}^{(n)}}}$ can be computed by the forward-backward algorithm similar to the one used for $Z_{\theta, \mathbf{x}, \mathbf{Y}}$. We then apply the standard optimization techniques to equation (2.12) as done in equation (2.10).

Chapter 3

Active Learning

Active learning is useful when manual labeling is costly. For example, labeling a part-of-speech of a word requires an expert [40]. In contrast, active learning may not be beneficial for easy labeling task such as image classification through World Wide Web [48] or labeling through a computer game [60].

Active learning is an interactive learning which selectively chooses training instances from the entire training set. In contrast to the passive learning, the selective sampling of active learning can reduce the annotation cost in terms of labeled instances and achieve high accuracy with the compensation of the training time.

3.1 Sampling Strategy

An early active learning approach is the membership query synthesis. A machine will ask an annotator to label the selected unlabeled instance and the synthesized instance [3]. This approach is not practical in many real world applications because human annotators have difficulties labeling the synthesized instance. More recent approaches are fallen in 2 categories, the stream-based selective sampling and the pool-based sampling strategies. Both approaches ask an annotator to label the real data.

Stream-based sampling assumes that the training instance is given one by one to the learning system, and will decide whether or not the given instance has to be labeled [14; 16]. On the other hand, the pool-based active learning will directly select the training instances from the training pool [29]. The decision in stream-based sampling, and the

Algorithm 1 Active learning framework

$S_t : \{(\mathbf{x}, \mathbf{y})\}$ is a set of all training sequences at iteration t
S_{sel} is a set of informative instances
$curmodel \leftarrow train(S_1)$ {Initial training}
repeat
  $S_{sel} \leftarrow Q(curmodel, S_t)$ {Sampling q instances}
  for $x \in S_{sel}$ **do**
    $S_{t+1} \leftarrow S_t \cup update(S_t, label(\mathbf{x}))$ {Annotation}
  end for
  $curmodel \leftarrow train(S_{t+1})$ {Training}
until stopping criterion is satisfied

selection in pool-based sampling are based on the informativeness of the instance. This thesis is concerned with the pool-based sampling strategy.

Pool-based sampling active learning framework is roughly divided into 3 phases, the sampling of informative instances, the annotation, and the re-training phases, as shown in Algorithm 1. The framework starts from a small initial labeled set. In each iteration, a model is trained using the current labeled set and is used to sample a collection of informative instances. An annotator is then asked to provide labels to the selected instances. Newly labeled data is added to the training set. The sampling and annotation are repeated until the stopping criterion is satisfied.

3.2 Informative Instances

An informative instance is an instance which can contribute to the desired result such as high accuracy, if it is annotated and added to the training set. The key idea of active learning is that some instances in the training data are more informative that the others. Active learning will choose only informative instances for labeling; hence reducing the annotation cost from the entire training set.

The definition of the informativeness based on the contribution to the desired result is hard to measure. Instead, we approximate the contribution by other measurements.

Let ϕ be the informativeness scoring function. This section provides two general scoring functions which are widely used in active learning, the uncertainty score and

the disagreement-among-committee score.

3.2.1 Uncertainty in the Prediction

The most popular informativeness measurement is the uncertainty of the prediction [44]. Given a single classifier, the classifier can predict output of an instance with some level of confidence. For a probabilistic model such as conditional random fields, the prediction confidence is in the form of output probability [15; 53]. For an input instance x, its informativeness in a probabilistic model is

$$\phi(x) = 1 - P(y^*|x) \ , \tag{3.1}$$

where y^* is the output label with the highest probability. When the output probability is high, it states that the model are confident in the prediction.

For a non-probabilistic model, we may obtain the probabilistic confidence by applying the sigmoid function [37]. For example, a maximum margin classifier can predict the output using the margin from the separating hyperplane. The margin score itself is a real value. Although we can normalize the margin score using the sigmoid function, the margin itself can be the prediction confidence [10; 41; 43; 55]. Let $f(x, y)$ be the predicted margin. The informativeness score of x is

$$\phi(x) = -|f(x, y^*)| \ . \tag{3.2}$$

When the absolute margin is large, it states that the model prediction is reliable.

Apart from the uncertainty of the candidate with the highest prediction confidence, the difference of confidence among the predictions is another measurement of uncertainty [9; 62]. We assume that a token will have a single output. For example, in the case of a probabilistic model, the informativeness based on the confidence difference is

$$\phi(x) = -(\max_{y' \in Y} P(y'|x) - \max_{y'' \in Y; y'' \neq y'} P(y''|x)) \ , \tag{3.3}$$

where y' and y'' are output labels in the set of all labels Y. When the difference between the prediction confidence is small, it indicates that the classifier cannot distinguish the best output from the other candidates. That instance will be highly informative.

Entropy of the prediction confidence is another measurement of uncertainty [46]. The informativeness score based on the entropy is defined as

$$\phi(x) = \sum_{y' \in Y} P(y'|x) \cdot \log(P(y'|x)) \ . \tag{3.4}$$

Low entropy represents the status that the classification is certain.

High prediction confidence or low uncertainty result implies that training corpus contains enough information for the classifier to distinguish the predicted output from the other outputs. Hence, an instance with high-prediction confidence is uninformative and is not required to be annotated. On the other hand, when the prediction confidence is low, it indicates that the training corpus lacks information to predict the output of the given input. An annotator should provide such information to the machine by annotating the instance. Therefore, such instances are informative.

3.2.2 Query by Committee

Instead of using a single classifier in the sampling strategy, we can use several classifiers with different hypotheses and evaluate the agreement among the prediction of classifiers [45]. There are two steps in the query by committee framework, the committee selection and the agreement measurement.

A single classifier is based on a single training set and single hypothesis. When a committee or a set of classifiers are necessary, we should either create several training sets or hypotheses. For the training set creation, we can randomly sample instances to make several subsets of the entire training set. Then, we can train a classifier or a committee on each subset. This method is called *query by bagging* [1]. For the creation of hypotheses, we can randomly set the model parameters according to some posterior distributions $P(\theta|L)$. For example, McCallum and Nigam sampled the naive Bayes parameters from a Dirichlet distribution [33]. Dagan and Engelson sampled HMM parameters from a normal distribution [16].

For the agreement measurement, Dagan and Engelson proposed the voting entropy which can measure the agreement among the prediction of the committee [16]. Supposing that there are C committee members, the voting entropy of the predicted labels is

$$\phi(x) = -\sum_{y' \in Y} \frac{V(y', x)}{|C|} \log \frac{V(y', x)}{|C|}, \tag{3.5}$$

where $V(y', x)$ denotes the number of committee members that predict the output of x to be y'. High entropy is interpreted as low agreement in the prediction. Thus, such instances are highly informative.

McCallum and Nigam proposed to measure the similarity between the prediction of committee members and the average prediction of all committees using the Kullback-Liebler (KL) divergence [33]. If the probability of the prediction of a committee member is $P_c(y|x)$ and the average probability is

$$P_{avg}(y|x) = \frac{1}{C}\sum_{c} P_c(y|x), \tag{3.6}$$

the KL divergence between these probability distributions is

$$D(P_c(y|x)||P_{avg}(y|x)) = \sum_{y' \in Y} P_c(y'|x) \log \frac{P_c(y'|x)}{P_{avg}(y'|x)} \ . \tag{3.7}$$

An instance with large divergence indicates that there is high disagreement among the predictions. Therefore, the informativeness based on the KL divergence is

$$\phi(x) = \frac{1}{C}\sum_{c} D(P_c(y|x)||P_{avg}(y|x)) \ . \tag{3.8}$$

3.3 Initial Training Set

A typical initial set may start from a collection of random instances. However, iterative methods are sensitive to the initial set, especially in the case of sparse data [24]. A reasonable initial set usually leads to convergence with less annotation cost.

A popular idea is to increase the coverage of the initial set as much as possible. Clustering-based methods are one of the famous selection strategies [24]. A representative instance of each cluster is selected for the initial set. Therefore, the initial set will have high coverage of the entire training set. The commonly used clustering algorithm is k-Means and its variations [24; 64]. Hu et al. proposed further-first-traversal, agglomerative hierarchical clustering, and affinity propagation clustering, which are deterministic algorithms, for the initial set selection of active learning [23]. They have pointed out that the k-Means and its variations are non-deterministic approaches, and are inconsistent. The deterministic approaches are more stable and can guarantee the same performance on every run.

In a structured output prediction problem, a large instance is supposed to contain more information than a small instance. Thereofre, a simple hueristic such as always choosing large instances is also effective [44].

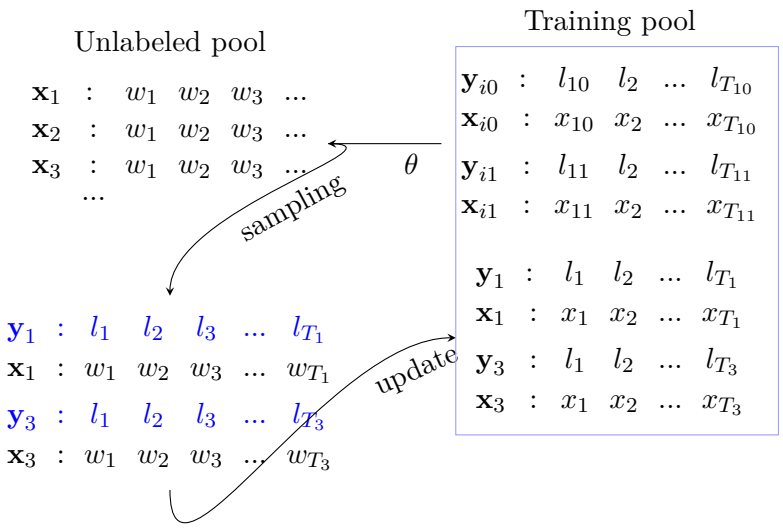


Figure 3.1: A general pool-based active learning framework for sequence labeling task

3.4 Stopping Criteria

An ideal stopping criteria is to finish with the highest accuracy, with the lowest annotation cost. In other words, the annotator should not add more annotated instances if those instances will not increase the accuracy. However, we do not know the accuracy of the system in advance.

We may limit the percentage of data to be annotated, or number of the iterations as a stopping criterion. The appropriate number depends on the task [59]. Another stopping criterion is to use the development set to evaluate the accuracy of the classifier [30]. However, this method requires an evaluation set to be annotated beforehand.

Vlachos proposed to evaluate the stopping criterion on an evaluation set, but they did not evaluate the classifier using the accuracy. Instead, they use the prediction confidence which does not require the annotated data [59]. We can also set the confidence threshold to be the stopping criterion, and stop when there is no informative instance according to the threshold [43].

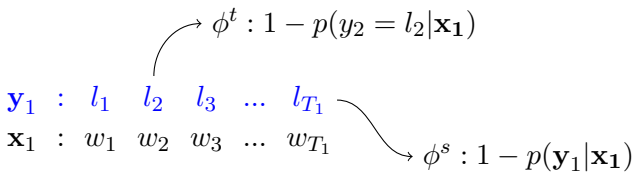


Figure 3.2: Sequence and token score

3.5 Active Learning for Sequence Labeling

The informativeness scores defined in Section 3.2 are the scores for a simple classification that has a single output for one input. When active learning is applied to a sequence labeling task, an input is an input sequence, while an output is an output sequence. Figure 3.1 illustrates an active learning for a sequence labeling task.

In the structured output problem, the output consists of several substructures. Moreover, the prediction is usually done at the substructure level. For the sequence labeling task, an input sequence structure consists of tokens. The output is also a sequence of label tokens. When we define the informativeness of an instance, we can either define it in the structure or substructure level as shown in Figure 3.2. In other words, we can calculate both the token score and the sequence score.

In the uncertainty-based sampling, we can simply use the probability or the margin of the sequence analogously to the simple classification. For the token score, we can use the either the marginal score [53] or the best prediction score [41]. However, the selective sampling is done at the sequence level. Therefore, we still need to merge the token scores to build a sequence score. One simple merging solution is to normalize the scores of all tokens in the sequence [4]:

$$\phi^s(x) = \frac{1}{T}\sum_{t=1}^{T}\phi^t(x) \ , \tag{3.9}$$

where ϕ^t and ϕ^s are the token score and the sequence score, respectively. T is the length of the sequence. The normalization is intended to reduce the bias on long sequences. However, Settles and Craven have argued that a long sequence usually contains more information than a short sequence and is more informative than a short sequence [44]. Therefore, the sampling should be biased to long sequences. The sequence score may

not have to be normalized by the sequence length:

$$\phi^s(x) = \sum_{t=1}^{T} \phi^t(x) \ . \tag{3.10}$$

For the label entropy in the uncertainty-based sampling, the straightforward calculation will be

$$\phi^s(x) = -\sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x}) \log P(\mathbf{y}'|\mathbf{x}) \ . \tag{3.11}$$

Since the summing over all possible $\mathbf{y}$ is intractable, Kim et al. proposed an approximation of the calculation using the N-best output [26]:

$$\phi^s(x) = -\sum_{\mathbf{y}' \in N} P(\mathbf{y}'|\mathbf{x}) \log P(\mathbf{y}'|\mathbf{x}) \ , \tag{3.12}$$

where N is the set of N best predictions.

Similarly, the sequence voting entropy and the sequence KL divergence of the query by committee strategy are also approximated using the N^c-best output. N^c is the union of N-best outputs from all committee members. The sequence entropy becomes

$$\phi(x) = -\sum_{\mathbf{y}' \in N^c} \frac{V(\mathbf{y}', \mathbf{x})}{|C|} \log \frac{V(\mathbf{y}', \mathbf{x})}{|C|} \ . \tag{3.13}$$

The KL divergence is slightly changed to

$$D(P_c(\mathbf{y}|\mathbf{x})||P_{avg}(\mathbf{y}|\mathbf{x})) = \sum_{\mathbf{y}' \in N^c} P_c(\mathbf{y}'|\mathbf{x}) \log \frac{P_c(\mathbf{y}'|\mathbf{x})}{P_{avg}(\mathbf{y}'|\mathbf{x})} \ . \tag{3.14}$$

Since a structure consists of several substructures, the sampling and annotation can also be done in the substructure level as illustrated in Figure 3.3. In the token annotation framework, only the informative tokens will be manually annotated. The other tokens can be labeled by the model estimation [53; 56]. In the substructure sampling, the entire structure is provided to an annotator because the context is important in the annotation. However, an annotator needs to label only the specified substructure. Sassano and Kurohashi proposed to sample a chunk rather than a sentence in Japanese dependency parsing task [42]. Morevoer, Druck et al. proposed the sampling and the annotation of the features instead of the output label [18].

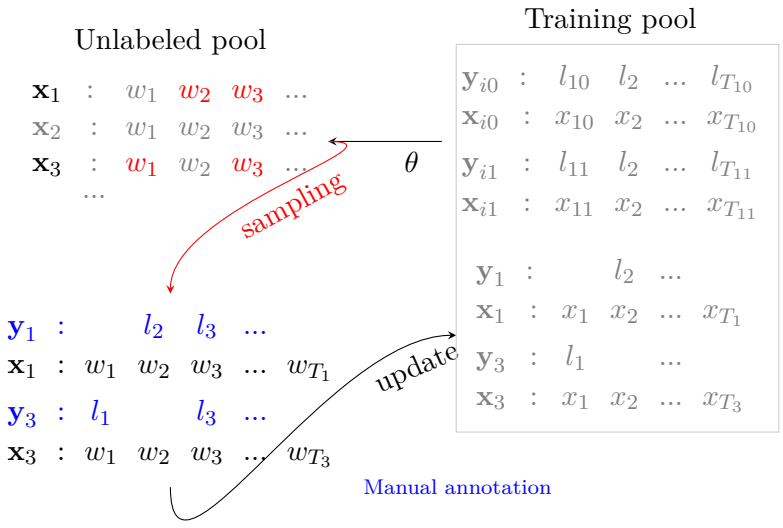


Figure 3.3: Partial sampling and partial anntoation

Chapter 4

Related work

Semi-supervised learning exploits the highly-accurate-but-costly labeled data, and noisy-but-cheap unlabeled data. Self-training is the simplest semi-supervised technique where we create the pseudo-labeled data from unlabeled data using the model trained on the labeled set [63]. However, we can achieve little improvement by this method especially when we have a small amount of labeled data. Since we assume that the unlabeled set has the same distribution as the labeled set, we can learn little new information from the unlabeled data. Co-training [8] uses multiple classifiers to create the pseudo-labeled data. When the agreement among classifier predictions is high, the pseudo-labeled instance is highly reliable [13]. The performance of both the self-training and the co-training approaches depends on the quality of the initial classifiers since the accuracy of the pseudo-labeled instances depends on these models.

When we have both labeled and unlabeled data, we can separately train a tagger from each type of data and combine them together using ensemble methods [35]. Ando and Zhang utilized unlabeled data by alternate structure optimization [2]. They annotated the unlabeled data using a set of simpler classifiers, and trained a new classifier using the labeled and automatically labeled sets. Suzuki and Izosaki proposed to train a tagger using both labeled and unlabeled data at the same time but assigned each type of data with its appropriate parameters [49]. Raina et al. proposed a self-taught learning approach that learns the general concept from unlabeled data and refines the tagger using labeled data [38].

The difficulty of labeling is often caused by the dependency between substructures. Obtaining partially labeled data in structural learning is easier than obtaining the fully

labeled data in some tasks. In the machine-aided corpus annotation, the annotator needs to verify the machine prediction. In many cases such as named entity recognition task, the entity is only a part of the whole structure. Tsuboi et al. proposed to automatically extract the entity parts through dictionary lookup [56], while Tsuruoka et al proposed to extract the entity parts using a classifier [57].

Although we know only the information of a part of the sample, we can find the information of the other parts from other samples. Therefore, we can also train a model on partially labeled data. Tsuboi et al. proposed to estimate the labels of unlabeled data using their marginalized probabilities [56]. Spreyer and Kuhn proposed to project the parse information from a source language to a target language in the parsing task [47]. Although the projected tree is usually incomplete, we can still train a parser using partially labeled trees. Sassano and Kurohashi proposed to train a parser for Japanese using a constituent rather than a sentence by utilizing the right-headed projective structure of Japanese language [42]. The idea is also applicable to the domain adaptation task where different domains share some general information. We only need to provide the domain-specific information from the target domain data, and gather the general information from the source domain data.

The current model may already be able to predict the correct output of some samples. In other words, these samples contain little new information for the training. We define such samples as *uninformative* samples and limit the annotation effort on these samples. Dasgupta and Ng proposed an active learning framework which could selectively sample informative instances for manual labeling and automatically label uninformative "easy" samples using a clustering approach [17]. We can measure the informativeness of a sample in several ways. Settles and Craven have analyzed several active learning strategies for fully labeled sequence labeling task [44]. For a structural sample which consists of substructures, the model may already be able to predict a part of the sample with high accuracy. Tomanek and Hahn applied this idea to reduce the annotation cost from the entire structure to partial structures by changing the labeling level from the sequence level to the token level [53]. They manually label the informative tokens, while they automatically label the uninformative tokens using the model predictions.

Bootstrapping is closely related to active learning in the way that new samples are selected and added to the training set in each iteration. In contrast to active

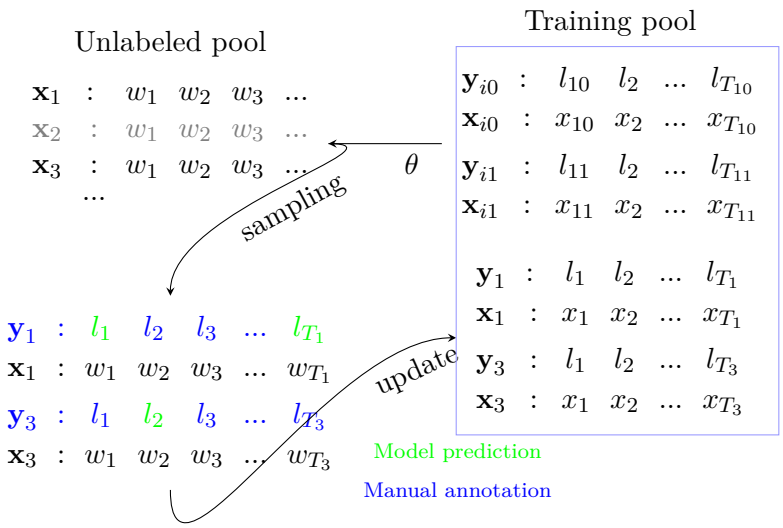


Figure 4.1: *[Tomanek09]*

learning, bootstrapping requires no human annotation effort. Therefore, bootstrapping strategy will select new samples whose prediction confidence is rather high to avoid labeling errors [6; 61]. Although bootstrapping requires less annotation cost than active learning, the performance of bootstrapping is still far poorer than that of the active learning, which exploits expensive annotated data.

4.1 Selected Baselines

We choose 3 pieces of related work which are closely related to our approach as baseline systems.

The first system is a semi-supervised active learning system of Tomanek and Hahn in [53]. They adopted a general pool-based active learning framework but slightly modified the annotation phase. They proposed to, automatically, label the uninformative tokens in the selected sequence by the model predictions. The other tokens, which are informative, are manually labeled. Therefore, they can reduce the annotation cost from the entire sequence to a part of the sequence. However, they did not incorporate the prediction re-estimation in the labeling. After a token in the sequence is labeled, the

$$
\begin{array}{llllll}
\mathbf{y}_{i0} & : & l_{10} & l_2 & \ldots & l_{T_{10}} \\
\mathbf{x}_{i0} & : & x_{10} & x_2 & \ldots & x_{T_{10}} \\
\mathbf{y}_{i1} & : & l_{11} & l_2 & \ldots & l_{T_{11}} \\
\mathbf{x}_{i1} & : & x_{11} & x_2 & \ldots & x_{T_{11}}
\end{array}
\qquad
\begin{array}{llllll}
\mathbf{y}_1 & : & & l_2 & l_3 & \ldots \\
\mathbf{x}_1 & : & w_1 & w_2 & w_3 & \ldots \; w_{T_1} \\
\mathbf{y}_3 & : & l_1 & & l_3 & \ldots \; l_{T_3} \\
\mathbf{x}_3 & : & w_1 & w_2 & w_3 & \ldots \; w_{T_3}
\end{array}
$$

Fully annotated source domain data

Partially annotated target domain data

Figure 4.2: *[Tsuboi08]*

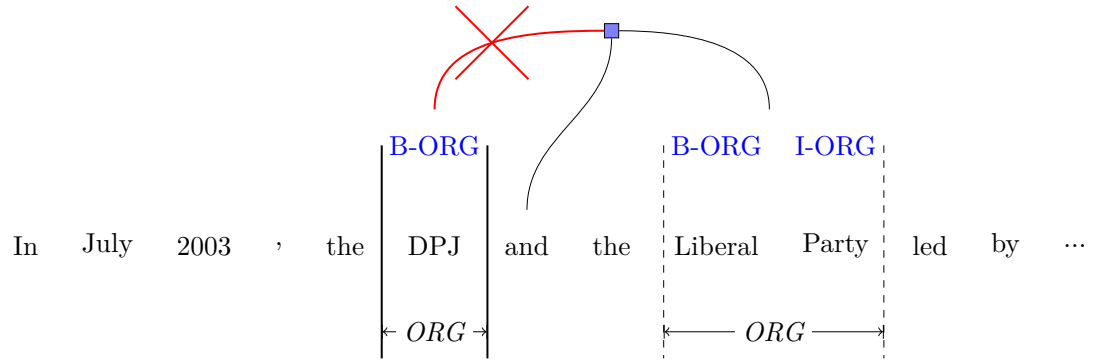


Figure 4.3: *[Neubig10]*

model prediction of the other tokens may change. Figure 4.1 illustrates the framework described in [53].

The second piece of work is a system of Tsuboi et al. which is trained on a partially annotated sequence [56]. They originally proposed a system for a domain adaptation task. Given the fully-labeled-source-domain data, the unlabeled-target-domain data, and an external dictionary, they automatically annotated words in the target domain data which are found in the dictionary, in order to obtain the partially annotated set. They finally trained a classifier using the fully-labeled-source and partially-labeled-target domain data as shown in Figure 4.2. While they automatically labeled the unlabeled sequences using an external dictionary, we propose a sampling method in active learning framework to choose words for manual annotation.

The last system is another training approach for a partially annotated sequence. Neubig and Mori proposed a point-wise training method which ignores the dependencies

among output labels [34]. In other words, they defined factors or features on only the input context. Figure 4.3 illustrates a feature without dependencies among output labels. Even though a sequence is partially labeled, we can simply train the classifier using only the labeled parts.

Chapter 5

Proposed Framework

Although an active learning framework can reduce the annotation cost by choosing only informative instances for labeling, there are still several rooms for further cost reduction in the sequence labeling problem. In this task, which is a structured output prediction problem, the labeling is factorized to the subsequence or token labeling. Therefore, we can also perform the active learning in the token level instead of the sequence level.

The informativeness of a token is defined by its prediction confidence. In this work, we employ the CRFs described in Chapter 2 for the training phase in the proposed framework. A CRFs classifier will predict an output label of a token with the following marginal probability:

$$P(y_j = y'|\mathbf{x}) = \frac{\alpha_j(y'|\mathbf{x}) \cdot \beta_j(y'|\mathbf{x})}{Z_{\theta,\mathbf{x},\mathbf{y}}} \ . \tag{5.1}$$

We utilize this marginal probability as the prediction confidence for a token. When the confidence is lower than a preset threshold, δ, the token may contain essential information which is not previously known to the model. Thus, we include this information to the model by manually putting the correct label on this token. As in Figure 5.1, the marginal probability of the first two tokens are 0.40 and 0.80, which are lower than the threshold, $\delta = .90$. Therefore, an annotator should manually label both tokens.

After a token is labeled, the prediction confidence of all tokens in the sequence will increase. In other words, the new information will propagate from the newly labeled token to the other neighboring tokens. If we re-estimate the confidence every time after the annotation of a token, we will substantially reduce the number of required labeled tokens.

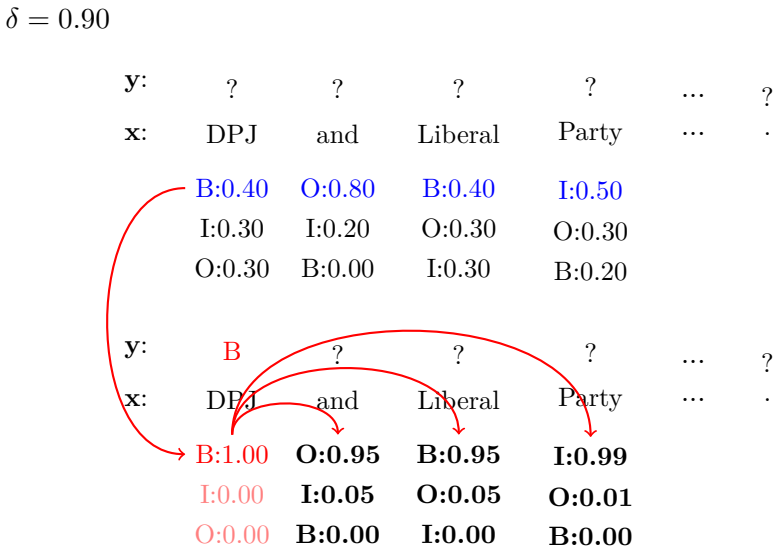


Figure 5.1: Marginal probability as the prediction confidence: The confidence threshold is set to 0.90. There are 3 label types, B, I, and O. The figure after the label is the output probability of each label. After a token is labeled, the probability of all tokens will change.

Algorithm 2 Proposed active learning framework with the confidence re-estimation

$S_{s,t}$: $\{(\mathbf{x}, \mathbf{y_L})\}$ is a set of all training sequences with current annotation at iteration t of subtask s
S_{sel} is a set of informative tokens
x is an input token
for each s **do**
  $curmodel \leftarrow train(S_{s,1})$ {Initial training}
  repeat
    $S_{sel} \leftarrow Q_{\text{tok}}(curmodel, S_{s,t})$ {Sampling q tokens (at most): Section 5.3}
    for $x \in S_{sel}$ **do**
      $S_{s,t+1} \leftarrow update(S_{s,t}, x, label(x))$ {Annotation: Section 5.1}
    end for
    $curmodel \leftarrow train(S_{s,t+1})$ {Training: Section 5.2}
  until $(|S_{sel}| < q)$ and $(\kappa(S_{s,t}, S_{s,t+1}) > stop)$ {Stopping criterion: Section 5.5}
  $S_{s,final} \leftarrow S_{s,t+1}$
end for
$S_{final} \leftarrow merge(S_{s,final})$ {Merge training sets: Section 5.4}
$finalmodel \leftarrow train(S_{final})$

$\mathbf{x}_1$: w_1 w_2 w_3 ... w_{T_1}
$\mathbf{x}_3$: w_1 w_2 w_3 ... w_{T_3}
...

(a) Given input sequence

$\mathbf{y}_1$: l_2 ...
$\mathbf{x}_1$: w_1 w_2 w_3 ... w_{T_1}
$\mathbf{y}_3$: l_3 ...
$\mathbf{x}_3$: w_1 w_2 w_3 ... w_{T_3}
...

(b) 1^{st} annotation

$\mathbf{y}_1$: l_2 l_3 ...
$\mathbf{x}_1$: w_1 w_2 w_3 ... w_{T_1}
$\mathbf{y}_3$: l_1 l_3 ...
$\mathbf{x}_3$: w_1 w_2 w_3 ... w_{T_3}
...

(c) 2^{nd} annotation

$\mathbf{y}_1$: l_2 l_3 ...
$\mathbf{x}_1$: w_1 w_2 w_3 ... w_{T_1}
$\mathbf{y}_3$: l_1 l_3 ... l_{T_3}
$\mathbf{x}_3$: w_1 w_2 w_3 ... w_{T_3}
...

(d) 3^{rd} annotation

$\mathbf{y}_1$: $\mathbf{l_1}$ l_2 l_3 ... $\mathbf{l_{T_1}}$
$\mathbf{x}_1$: w_1 w_2 w_3 ... w_{T_1}
$\mathbf{y}_3$: l_1 $\mathbf{l_2}$ l_3 ... l_{T_3}
$\mathbf{x}_3$: w_1 w_2 w_3 ... w_{T_3}
...

(e) Final annotation

Figure 5.2: Confidence re-estimation in the annotation. An annotator labels on token in the sequence in each step ((b) to (e)). At the final iteration, low-informative tokens (in bold figures) are labeled by the model prediction.

We adopt the general active learning framework in Chapter 3 and integrate the confidence re-estimation in the annotation (Section 5.1), the training (Section 5.2), and the sampling phases (Section 5.3). The outline of the proposed framework is shown in Algorithm 2. We also propose the simplified sampling to reduce the computational training time (Section 5.4) and discuss the stopping criterion in Section 5.5.

5.1 Confidence Re-estimation in the Annotation

We assume that the human annotation is always correct. When a token is manually annotated, the classifier will select the human annotation as the output label. Hence, the prediction probability of the labeled token is always 1.0. We propose to re-estimate the confidence after every single token annotation. An example of a sequence annotation is shown in Figure 5.2.

Given the input sequence in Figure 5.2a, the most informative token in the sequence is selected and labeled by the annotator as shown in Figure 5.2b. Subsequently, the

prediction confidence of all tokens in the sequences is re-estimated. Then, the next highest informative token is selected and labeled until there is no informative token left in the sequence. Finally, all of the uninformative tokens are labeled by the model prediction. Therefore, we can obtain a labeled sequence for training. However, the annotator only puts labels to some tokens in the sequence. Figure 5.2e shows the labeled sequence for the training. Labels in boldface are automatically annotated, while the other tokens are manually annotated.

5.2 Confidence Re-estimation in the Training

The conventional CRFs described in Section 2.2 require a sequence to be fully annotated. We can annotate sequences using both the human annotation and machine annotation as done in Section 5.1. However, the accuracy of the models in early iterations is usually low, which makes the prediction confidence from those early models not reliable. In addition, adding these incorrectly predicted outputs to the training set will prevent the model from achieving high accuracy. These errors remain in the training data and are not recovered, even though the classifier becomes more accurate in later iterations.

We propose to re-estimate the output prediction in every iteration, instead of fixing labels to the prediction of the model in early iterations. When the classifier becomes more accurate in later iterations, the prediction will also be more accurate. We directly estimate the prediction in the training phase using CRFs for partially annotated sequence proposed by Tsuboi et al. in [56]. They proposed to fill the unlabeled tokens with their marginalized output. In other words, we implicitly annotate the unlabeled tokens with all possible output labels, augmented with their prediction probabilities. As a result, when the model becomes more precise, these implicitly annotated tokens will also be more precise.

Figure 5.3 shows an example of the re-estimation in the training. We sample and annotate tokens similar to the process done in Section 5.1. The difference is in the final annotation. We do not put the predicted labels on high-confidence tokens. Instead, we use the implicit annotation through the re-estimation in the training phase.

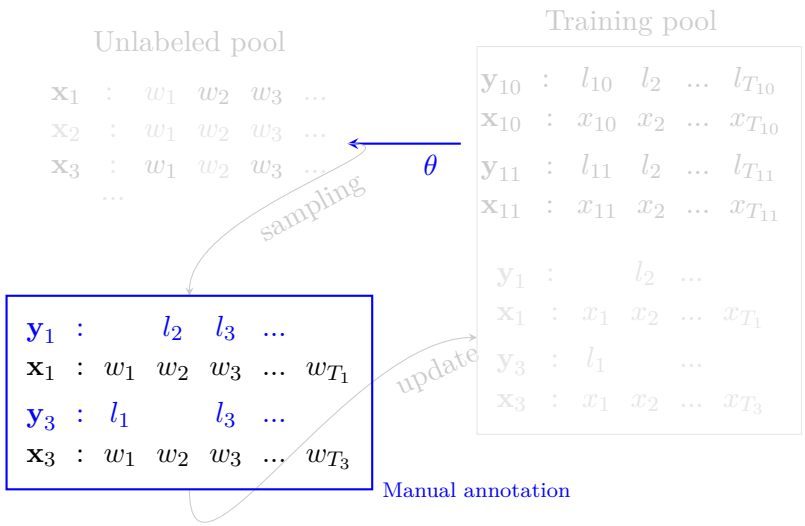


Figure 5.3: Confidence re-estimation in the training. An annotator labels one token in the sequence in each step ((b) to (e)). No token is explicitly labeled by the model prediction.

5.3 Token Sampling Strategy

We have discussed that the accuracy of the classifier has an effect to the reliability of the prediction confidence. We can lessen the impact of a poor classifier from early iterations by re-estimating the prediction in the training process. In this section, we argue that the model accuracy also affects the sampling process, since the sampling also depends on the prediction confidence. Desired sampling should be able to retrieve many informative tokens. Therefore, we can expect less efficient sampling results from early models than from those in later iterations.

When many tokens in a sequence are annotated, the probability of the other unlabeled tokens is generally high. If the labeled tokens are uninformative or less-informative tokens, we will miss the chance to annotate other informative tokens. Therefore, we need to re-train the model every time after a single token in a sequence is annotated. However, the training cost will be extremely high. We, instead, propose to sample a set of sequences and annotate only the highest informative token in each sampled sequence. Specifically, we sort all tokens in the corpus by their confidence

scores in ascending order. In other words, we perform token sampling instead of sequence sampling, in order to utilize the probability re-estimation in the sampling. A sequence may be sampled several times if there are several informative tokens in the sequence.

5.4 Simplified Sampling

Active learning can reduce the annotation cost in terms of labeled tokens at the expense of additional computational time. The re-estimation in the annotation and the training phases does not require high computational time. On the other hand, the re-estimation in the sampling, which includes both the training and re-estimation for one iteration, requires high computational cost. The training phase is the most time consuming process in the whole framework. Since the training is the tuning of the model parameters and one parameter corresponds to one feature, the training time depends on the number of features, or factors, in the training set.

Let the corpus have N unique words and L unique labels. A feature is a relationship between arbitrary inputs and outputs. We also augment all features with the output label of the given input. Therefore, the number of features depends on the size of N and L. For example, a trigram word feature has a size of $L \cdot N^3$. If the size of L is reduced, the number of features in the corpus will decrease. Consequently, the training time should also decrease.

The original corpus contains various types of entity tokens. For example, a named-entity corpus may have *PERSON*, *PLACE*, *ORG*, and *MISC* entity types. We propose to split the labeling task into a set of single-entity labeling tasks, i.e. a *PERSON* labeling task, etc. We employ the active learning with confidence re-estimation to label each sub task. The final corpus of each sub task is a partially annotated corpus. These corpora are merged to create the final corpus, which is still partially labeled. The final classifier is trained on the merged corpus.

We assume that there is a single label for one token, and no ambiguity from human labeling. Therefore, a token which is labeled as an entity type in a sub task, should be labeled as a non-entity type in the other sub tasks. For example, a token labeled with *PERSON* entity type should not be labeled with *PLACE* at the same time. In contrast, a token labeled as a non-entity type in a sub task may be a real non-entity,

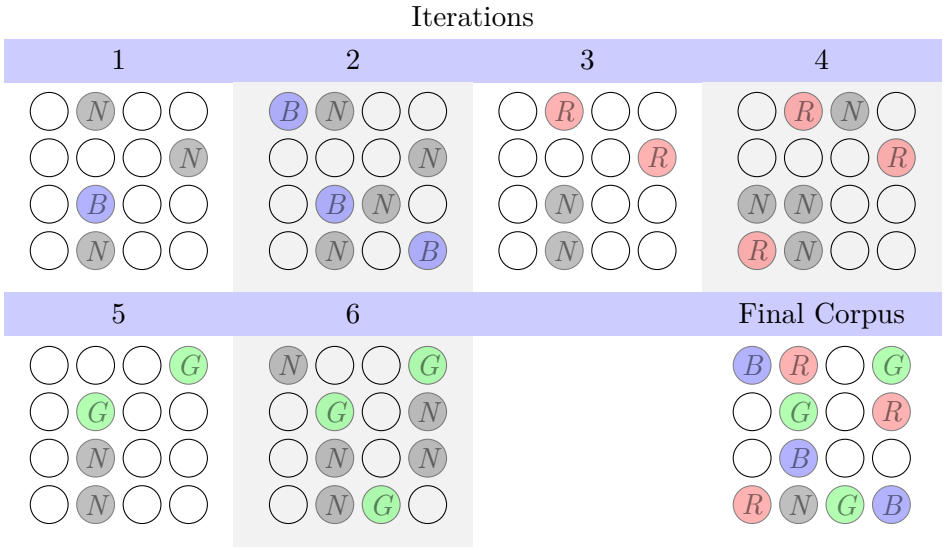


Figure 5.4: An example of simplified sampling

or an entity of the other types. A token is regarded as a non-entity token if and only if it is labeled as a non-entity type in all sub tasks. Otherwise, the token is left unlabeled in the merged corpus.

However, we found that few non-entity tokens are labeled in all sub tasks. Therefore, the merged corpus will contain mostly entity tokens, lacking non-entity tokens. This merged corpus is inappropriate for the training. We propose to add non-entity labels to the training set by using the model prediction of all sub tasks. There may be conflicts among the model predictions. In such cases, we leave the tokens unlabeled.

An example of the simplified sampling is shown in Figure 5.4. Let us have 4 class labels, the B, the R, the G entity classes, and the N non-entity class. Instead of asking an annotator to determine the class label of a token among 4 label candidates, we divide the tasks into 3 subtasks, the B labeling, the R labeling, and the G labeling. After all sub tasks are done, we merge the partially annotated data of each sub task together and train the final model using the merged data.

5.5 Stopping Criterion

The stopping criterion is also a prominent key of active learning to reduce the annotation cost by stopping the labeling when the accuracy converges to the desired level.

Firstly, we predict the output of all training sequences using the current model in each iteration. Note that the output of a labeled token is perfect, and always correct. Secondly, we measure the similarity between the prediction of the models from two consecutive iterations using Kappa statistic [11], which is a measure of inter-annotator agreement. The probability of two corpus annotation, A_o is

$$A_o = \frac{nAgree}{nAll} \ , \tag{5.2}$$

where $nAgree$ and $nAll$ are the number of prediction agreement, and the number of all predictions, respectively. Let the number of label l predictions in a corpus i be n_l^i. The probability that the agreement occurs by chance is

$$A_e = \sum_l \frac{n_l^1 n_l^2}{(\sum_l n_1^1)^2} \ . \tag{5.3}$$

We can calculate the Kappa statistic, κ, by the following equation:

$$\kappa = \frac{A_o - A_e}{1 - A_e} \ . \tag{5.4}$$

When κ is high, it indicates that the agreement does not occur by chance, and the two corpora are very similar. The learning can be stopped when κ between iterations are high enough. We empirically tuned κ and set the threshold to 0.9999. When κ exceeds the threshold, the labeling will be stopped.

Chapter 6

Experiments

6.1 Data and Evaluation

We evaluate the proposed framework on two tasks, the English all-phrase chunking and the English named-entity recognition tasks.

6.1.1 CoNLL2000

The first task is the English all-phrase chunking. We conduct the experiments using English all-phrase chunking data set in CoNLL2000 shared-task [50]. The objective of the chunking is to find all non-recursive chunks or phrases of all types in the given sentence. The all-phrase chunking task is also called a shallow parsing.

A sample sentence, *"Meanwhile, overall evidence on the economy remains fairly clouded."*, in Figure 6.1 contains 1 adverb phrase, 2 noun phrases, 1 prepositional phrase, 1 verb phrase, and 1 adjective phrase. The task is also formulated to be a sequence labeling task. A chunk is constructed of two types of labeled tokens, the starting token, and the other tokens inside the chunk. The tokens outside a chunk are labeled as non-chunk type. The labeling convention is denoted as IOB-2 format [52].

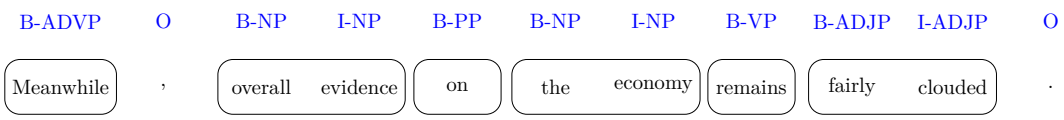


Figure 6.1: All-phrase chunking task

Table 6.1: CoNLL2000 data; Number of sequences, tokens, chunk types and chunks of all-phrase. NP chunking is the simplified task on the same data set.

Dataset	Num. Seq.	Num. Tok.	Num.Chunk types	Num. Chunks
All-phrase Chunking				
training	8936	211727	22	106978
test	2012	47377	19	23852
NP-Chunking				
training	8936	211727	3	55081
test	2012	47377	3	12422

Therefore, the output is a label sequence of the same length of the corresponding input sequence.

The data statistics of CoNLL2000 is shown in Table 6.1. We also simplify the data set to one-phrase, NP, chunking task in order to compare the performance of the proposed system between the single-entity and all-entity tasks.

We extract the features following [49] and employ a CRFs classifier to predict the output of each token. We set the context window to extract features and let the current word be the center of the window. All features are binary features. The list of features is as follows:

- Word n-gram: unigrams in the word window of size 5, bigrams in the window of size 3
- Part-of-speech n-gram: unigrams, bigrams and trigrams in the window of size 5
- Transition: label of the previous word

Note that the transition function makes the task a structured output problem. Without this feature, the token labeling becomes a simple classification task.

6.1.2 CoNLL2003

A named entity is a phrase that contains a name of person, organization, place, time, and quantity [51]. The named entity recognition task is the task to find all named entities in a given sentence. We perform the experiment using English data set from CoNLL2003 shared-task [51]. A named entity chunk is also constructed of two types of

Table 6.2: CoNLL2003 data: Number of sequences, tokens, chunk types and chunks of English NER task.

Dataset	Num. Seq.	Num. Tok.	Num. Chunk types	Num. Chunks
English NER				
training	14987	204567	8	23499
test	3684	46666	8	5648

Table 6.3: Word types and examples

Description	Examples
starts with capital letter	Confidence, September, But
all capital letters	PLC, GNP, A
contains both uppercase and lowercase letters	anti-American, Chancellor, Lawson
single digit	1, 2, 3
contains only digits	16, 1988, 190
contains at least two periods	U.K., A.P., F.S.B
ends with a period	Ala., p.m., vs.
contains a dash	year-ago, 1-800-453-9000, 10-fold
single character	A, a, b
contains punctuation	US$, #, /,/
contains quotation	's, I'm, n't

tokens. The data set uses slightly different representation of labels from the chunking data set. Instead of using the starting and in-chunk labels, all tokens are labeled as in-chunk tokens. Only if there are two consecutive entities, the starting token of the successor is labeled as the starting label. The representation is IOB-1 format [39].

Again, we extract features and predict the chunk label using a CRFs classifier. We set the context window to extract features and let the current word be the center of the window. All features are binary features. The list of features in this task is summarized as follows:

- Word n-gram: unigrams in the word window of size 5
- Previous and following word sequence: a 4-word sequence before and after the current word

Table 6.4: Prediction confusion marix

	is a chunk in the corpus	is *not* a chunk in the corpus
is a predicted chunk	tp	fp
is *not* a predicted chunk	fn	tn

- Lowercase n-gram: unigrams and bigrams of lowercase words in the window of size 5, a trigram word in the window of size 3
- Word type: types summarized in Table 6.3
- Part-of-speech n-gram: unigrams, bigrams, and trigram in the window of size 5. Part-of-speech information is provided in the data set.
- Chunk type n-gram: unigrams, bigrams, and trigram in the window of size 5. Chunk type, e.g. NP, is also provided in the data set.
- Character prefix and suffix: 2- and 3-character prefixes and suffixes of the input word. For example, *Ame* is 3-character prefix of the word *American*.
- Transition: label of the previous word

6.1.3 Evalutaion

The objective of the proposed framework is to achieve the supervised accuracy with as little annotation effort as possible. The supervised accuracy is the accuracy of a system trained on a fully annotated corpus. We measure the performance of each trained system by CoNLL chunk F_1 [50]. We use the word *accuracy* interchangeably with F_1 in this thesis.

A chunk is counted as correct if and only if the output of all tokens in the chunk are correctly predicted (tp). All other predicted chunks are counted as false predicted chunks (fp). Finally, the incorrectly predicted chunks are counted to be unrecognized chunks (fn). The rest are any tokens stay outside of a chunk (tn). We summarize these counts in the confusion matrix, in Table 6.4.

The precision (Pr) and recall (Re) are calculated by the following equations:

$$Pr = \frac{tp}{tp + fp} \ , \tag{6.1}$$

$$Re = \frac{tp}{tp + fn} \ . \tag{6.2}$$

The CoNLL chunk F_1 is the harmonic mean of the precision and recall:

$$F_1 = \frac{2(Pr)(Re)}{Pr + Re} \ . \tag{6.3}$$

We also evaluate the statistical difference of F_1 between systems using McNemar test [21]. The null hypothesis of the test states that the prediction errors of two classifiers are independent of each other. Let the errors from a classifier i be err_i. The McNemar test is given by

$$\chi^2 = \frac{(|err_1 - err_2| - 0.5)^2}{err_1 + err_2} \ . \tag{6.4}$$

We will reject the hypothesis, i.e. the difference is statistically significant, when χ^2 is higher than 3.84 with $\alpha = 0.05$.

We divide the annotation effort into 2 types. The first type is the number of tokens, or words, that a human annotator put a label for. We use the percentage of the annotated tokens in the whole corpus to represent this annotation cost. The other type is the computational time required in the full annotation task. Since active learning is an iterative method, a human is asked to annotate new tokens in every iteration. We assume that the annotation cost of each token is equal. We measure the idle time of the annotator in each iteration, also the cumulative idle time of the annotator[1]. Note that we did not consider the actual annotation time of a human annotator.

6.2 Parameter Tuning

6.2.1 Initial Training Set

We compared the F_1 of the proposed framework starting from the collection of short sequences, long sequences and random sequences in Figure 6.2. Each initial set consists of 50 fully annotated sequences. The confidence threshold is set to 0.99. New 500 most informative tokens are selected and manually labeled in each iteration of active learning. In Figure 6.2, long-sequence initial set leads to the supervised accuracy with fewer labeled tokens than when starting with the other initial sets.

[1]We conducted all experiments on 3.00GHz Xeon E5450 server using 4 cpus. Our CRFs classifier was implemented in C.

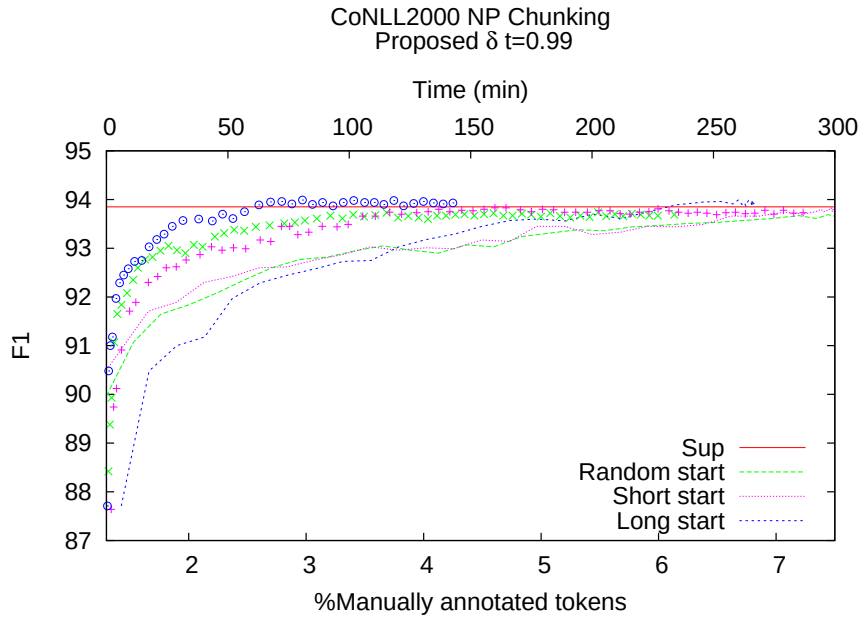


Figure 6.2: F_1 on the systems trained of different initial sets. Lines represent the number of labeled tokens. Points of the same color represent the CPU time.

Note that long sequences contain more tokens than short sequences. Although we assume that the annotation cost of all tokens is equal, an annotator should spend more annotation effort on the long-sequence initial set. However, the initial set contains only 50 sequences, which may not expect much annotation effort. Moreover, points on the top of Figure 6.2 show that the long-sequence initial set also achieves higher accuracy with similar CPU time. Hence, we will use 50 longest sequences as our initial set in all of the following experiments.

6.2.2 Sampling Size

Sampling size is the number of instances given to an annotator for labeling in one iteration. Sampling size is also another decisive factor in active learning. When the sampling size is large, some less informative instances are sampled for annotation. On the other hand, one extreme setting is to annotate one instance per iteration, but the CPU time will tremendously increase. We should consider these two costs, which become a trade-off of each other.

We perform the analysis using the proposed framework without the simplified sampling on NP chunking data set. The sampling size is set to 50, 500, and 1000 tokens

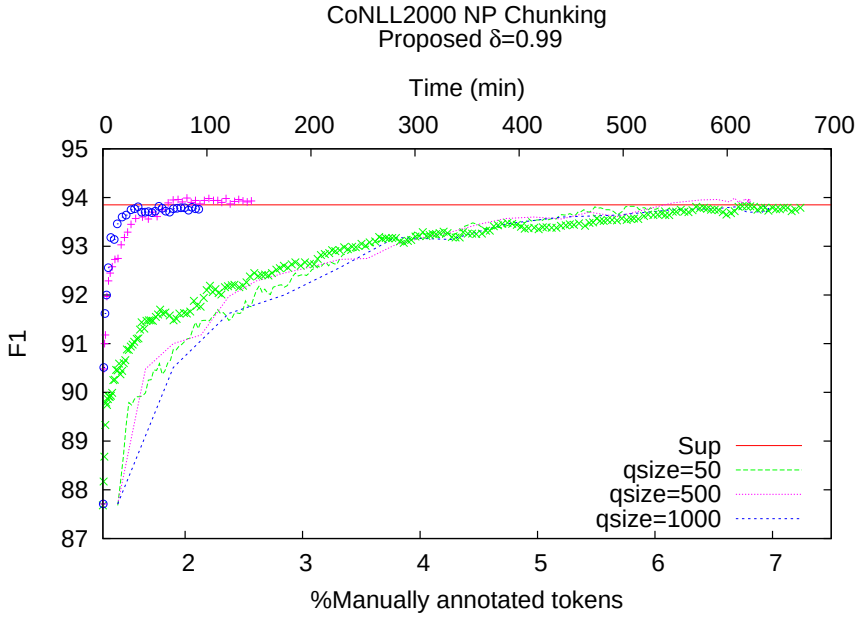


Figure 6.3: F_1 of the systems with different sampling sizes. Lines represent the number of labeled tokens. Points of the same color represent the CPU time.

per iteration. The lines in Figure 6.3 show that there is no significant difference in the F_1 achievement of individual sampling size settings. From the points of small sampling size setting, the setting significantly increases the idle time cost. We try to limit the idle time per iteration to several minutes. In the real annotation scenario, the idle time becomes a short break for a human, before starting labeling the next iteration. From the experiments, we found that the sampling size of 500 tokens is the most efficient.

6.2.3 Confidence Threshold

The last parameter is the prediction confidence threshold, δ, which defines the informativeness of a token. There is a trade-off between the high and the low-threshold settings. The reliability of the model prediction is high when the prediction confidence is high. Labeling all tokens in the selected sequence is equivalent to the threshold at positive infinity. On the other hand, the threshold at zero is equivalent to the self-training bootstrapping system. High-threshold setting will require a high number of labeled tokens. Although we can reduce the number of labeled tokens using a low threshold, we expect more risk of increasing incorrectly predicted labels to the training set, which leads to low-accuracy performance.

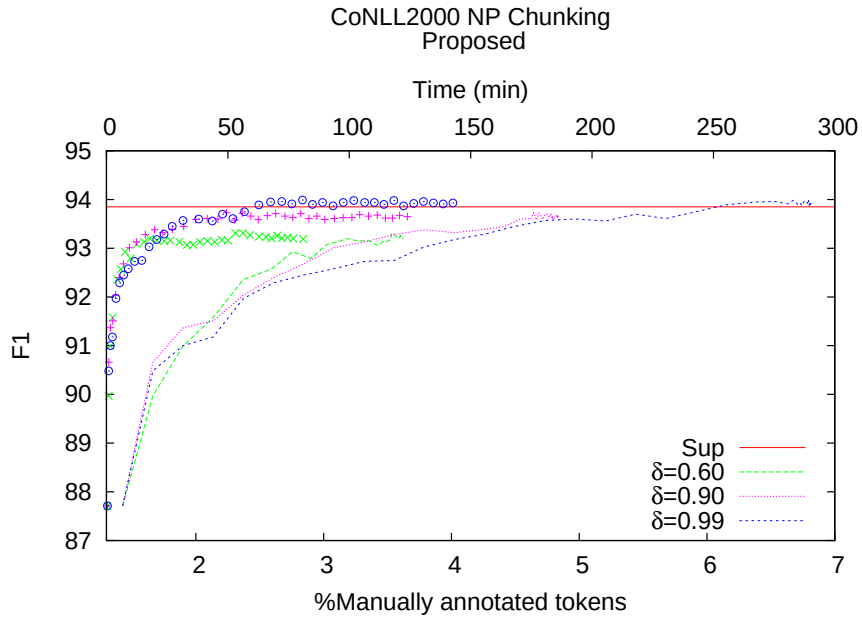


Figure 6.4: F_1 of the systems with different confidence threshold. Lines represent the number of labeled tokens. Points of the same color represent the CPU time.

We conduct the experiment using the re-estimation in the annotation, the training, and the sampling. Our data set is the NP chunking set. Lines in Figure 6.4 show, as expected, that the low-confidence threshold setting leads to fast convergence, and achieves low F_1. The difference of F_1 at the convergence of the system with the threshold at 0.90 and 0.99 is not statistically significant. The result also confirms that high-threshold setting may also lead to over-annotation, similar to the large sampling size case.

6.3 Baseline Systems

There are several pieces of work related to the proposed system. In this section, we will compare this related work before comparing them to the proposed system in the following sections.

[Sup]: The F_1 of this straightforward supervised system trained on a fully annotated corpus is the upper-bound of all systems. Our goal is to achieve this supervised F_1 with the smallest number of labeled tokens.

[Pointwise]: Neubig and Mori proposed a point-wise training framework in the

Table 6.5: Comparison between baseline systems on NP chunking task. Boldface figures are F_1 which are not statistically different from the supervised F_1.

Systems	F_1	%Labeled tokens	Training time (sec)
[Sup]	**93.86**	100.00	521
[Pointwise]	92.52	100.00	57
[Bootstrap]	89.05	1.43	1302
[AL]	**93.85**	100.00	4019
[Tomanek09] $\delta = 0.99$	92.91	5.48	3432

domain adaptation task, which ignores the dependencies among the output labels [34]. We emulate their system in a supervised manner. Specifically, we use all features except the transition feature. The supervised point-wise system will provide the upper bound accuracy of a point-wise active learning system.

The first half of Table 6.5 are the non-iterative systems. Both systems are trained on the same fully annotated corpus. However, *[Pointwise]* achieved significantly lower F_1 than *[Sup]*. Hence, we conclude that the transition features in *[Sup]* are responsible for the large gap between the two F_1 achievement. All of the following settings will include the transition features.

The other baselines are iterative sequence-sampling systems. All systems start with the same long-sequence initial set. The query size is set to 500 sequences per iteration. The confidence threshold, δ, is set to 0.99.

[Bootstrap]: The first iterative baseline is a bootstrapping system, whose high-confidence tokens are annotated with the model prediction. Labels of low-confidence tokens are implicitly estimated in the training.

[AL]: The second iterative baseline is a fully supervised active learning system, whose training tokens are manually labeled. *[AL]* becomes almost exactly the same as *[Sup]* after we label the whole training set. The only difference between *[AL]* and *[Sup]* is the sentence ordering in the training set.

[Tomanek09]: The last baseline is the system proposed by Tomanek and Hahn [53]. In each sampled sentence, the low-confidence tokens are manually labeled, while the high-confidence tokens are automatically labeled by the model prediction.

Figure 6.5 shows the correlation between the number of labeled tokens and F_1 of each baseline. Note that the F_1 of *[Sup]* does not vary since we train the tagger once with

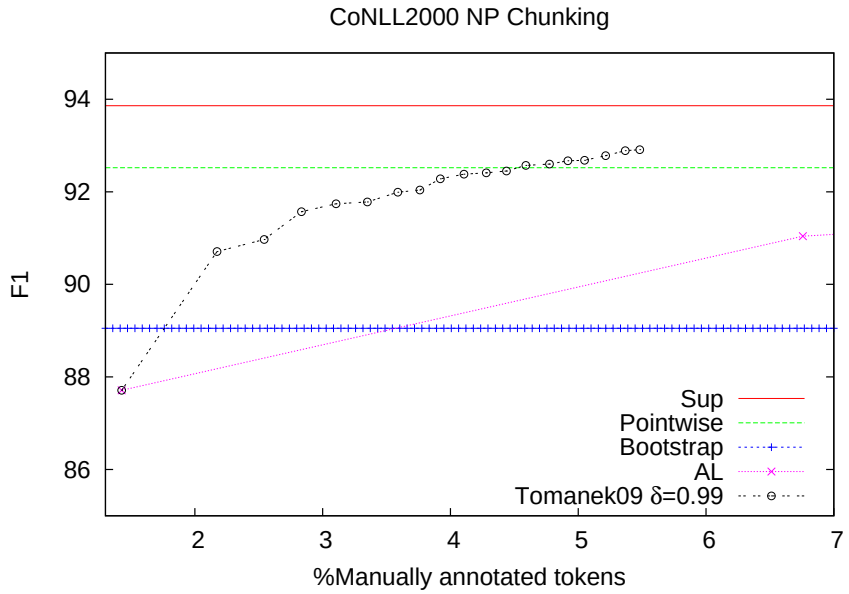


Figure 6.5: F_1 comparison among baseline systems

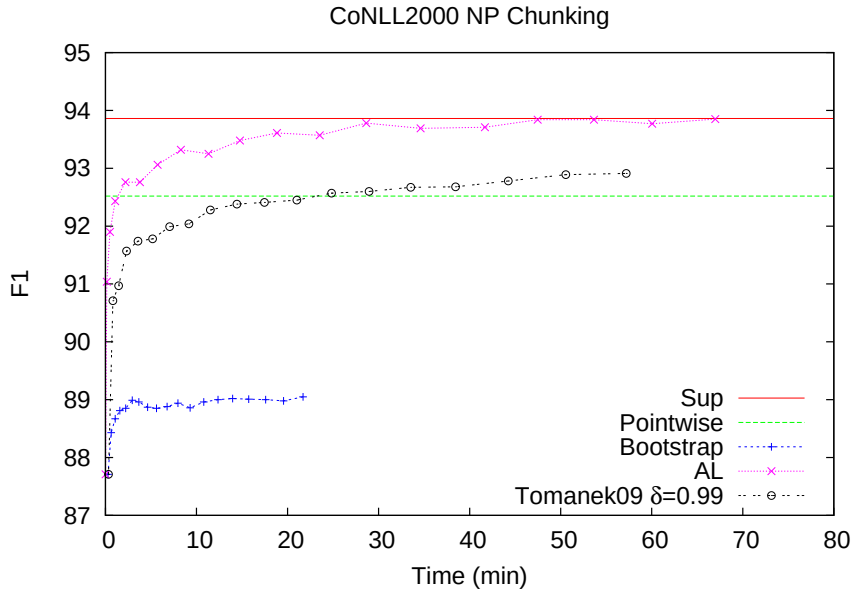


Figure 6.6: Training time comparison among baseline systems

the whole training set. We just draw a straight F_1 line of *[Sup]* for reference. Similarly, the number of manually labeled tokens of *[Bootstrap]* is also fixed to the number of tokens in the initial set. However, the F_1 of *[Bootstrap]* varies since we gradually add tokens to the training set. We draw a straight line of final F_1 of *[Bootstrap]* vs. number of labeled tokens.

The straight line at the bottom of the figure is the F_1 of the bootstrapping system (*[Bootstrap]*), which indicates that we can hardly learn new information from unlabeled tokens. Although the F_1 of *[Bootstrap]* slightly increases, its best F_1 is still far worse than any other baselines that include more labeled tokens.

There are two learning curves in the middle of Figure 6.5. The bottom curve is *[AL]*, which has a low-learning rate. We cut the learning curve to first 15k labeled tokens due to the limited space. However, the F_1 of *[AL]* gradually increased and reached the supervised level when nearly 150k of tokens are labeled. The other curve represents the learning rate of *[Tomanek09]*. Although *[Tomanek09]* achieved much higher F_1 than *[Bootstrap]* with 4% additional labeled tokens, its best F_1 is still far worse than the supervised F_1.

Figure 6.6 shows the training time of each baseline. The training time of few labeled tokens is rather fast. We will discuss the training time later in Section 6.7.

Finally, we compare the performance of iterative baselines to non-iterative baselines in Table 6.5. In terms of the number of manually labeled tokens, active sampling methods obviously require fewer labeled tokens and achieve higher F_1 than non-iterative methods. Since the model becomes more precise in later iterations, we need to label few tokens but still achieve high F_1. In contrast, the iterative methods have a disadvantage in the training time since we have to re-estimate the model parameters in each iteration. In this thesis, we give priority to the labeled token cost.

6.4 Confidence Re-estimation in the Annotation

In the annotation phase, the re-estimation can reduce the annotation cost on tokens in the following categories. The first type is the tokens which are correctly predicted by the model, but have low-prediction confidence. After the re-estimation, these tokens will gain more prediction confidence. When the confidence passes the threshold, we can reduce the annotation cost on these tokens but still have the correct prediction. The

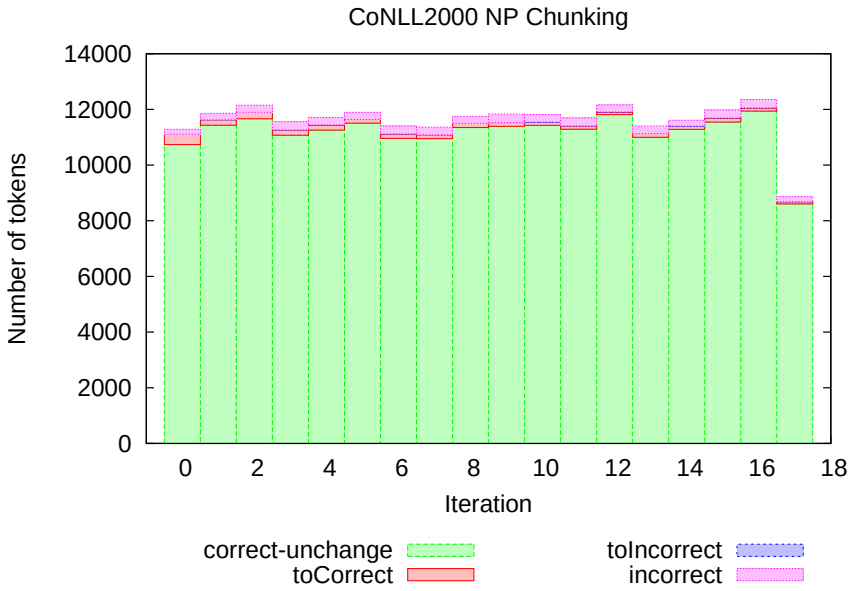


Figure 6.7: Label changes after re-estimation in the annotation phase

second type is the tokens with the outputs which are previously incorrectly predicted, but their outputs are corrected after the re-estimation.

From Figure 6.7, labels of the most tokens remain unchanged after the re-estimation. However, the prediction confidence of each token is increased. Therefore, we are required to label fewer tokens but still achieved the system with similar accuracy to the system without the re-estimation. The result is also shown in Figure 6.8 that the F_1 scores before and after re-estimation were not significantly different.

We also compute the kappa statistics between the corpora before and after the re-estimation. The agreement between the two corpora has $\kappa = 0.99$. This kappa value indicates that both annotated corpora are highly similar. We conclude that the re-estimation does not significantly alter the predicted output but raises the confidence of some tokens to exceed the threshold level. Therefore, we can reduce the annotation on these tokens. However, we cannot improve the F_1 of the system since the corpora before and after re-estimation are remarkably similar.

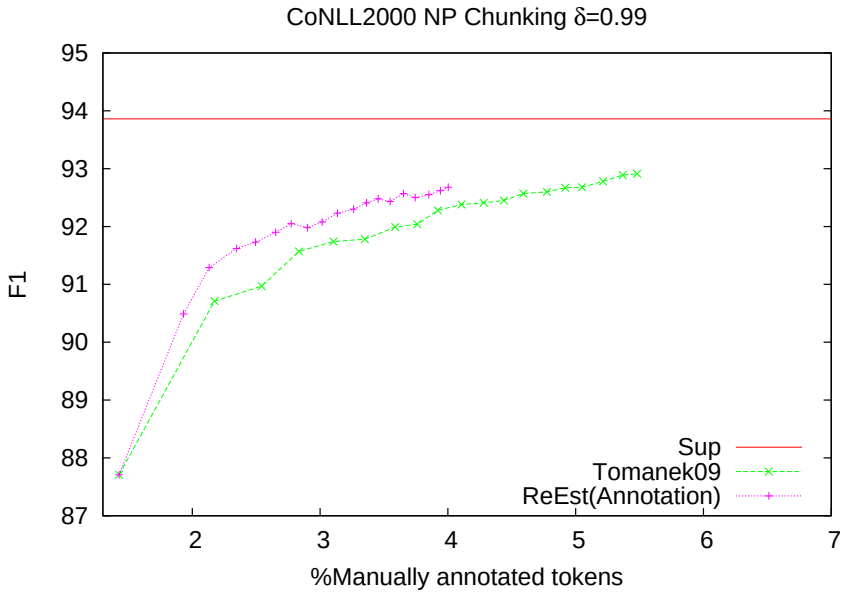


Figure 6.8: F_1 of the model with re-estimation in the annotation phase

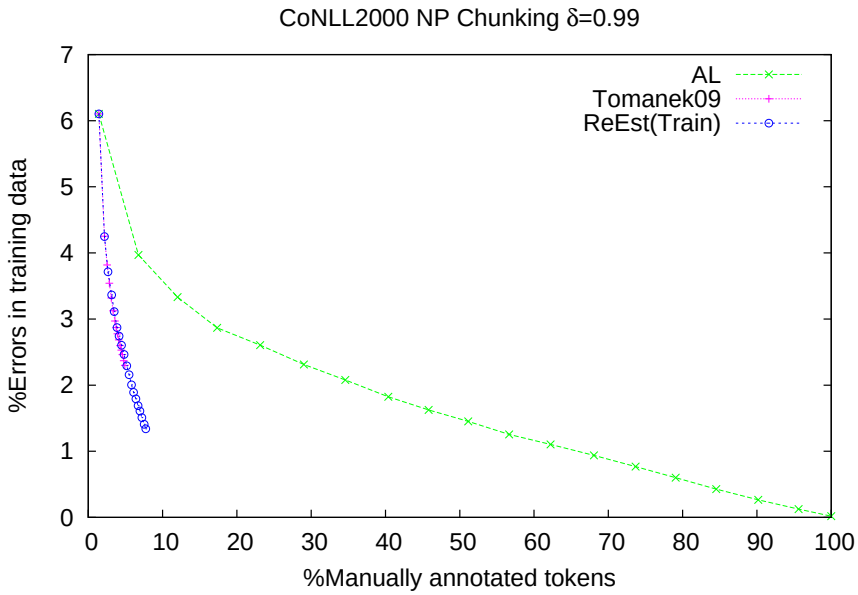


Figure 6.9: Errors in the training set with and without re-estimation

6.5 Confidence Re-estimation in the Training

We conduct an experiment using a fully annotated corpus as a gold standard, and evaluate the prediction accuracy of the model in each iteration on the entire training corpus against the gold standard. The accuracy of a labeled token will be 1.0. The accuracy of the other tokens depends on the prediction confidence. If an incorrectly predicted label is added into the training set, the accuracy of that token becomes zero. On the other hand, if we estimate the label in training, the error will depend on the prediction confidence. The result in Figure 6.9 shows that every setting starts with the same error level of the same initial set. When more tokens are labeled, the errors decrease because of the labeled tokens themselves and the improved accuracy of the model. Errors of *[AL]* continue decreasing to zero when all tokens are annotated. In contrast, errors of *[Tomanek09]* sharply decrease but stop early while errors are still high. Therefore, the model trained on this erroneous training corpus achieved lower F_1 than the fully supervised model.

The proposed framework, which implicitly estimates the output labels in the training, does not suffer much from the incorrect prediction. Although the errors are high in early iterations similar to *[Tomanek09]*, the prediction is automatically corrected after a model becomes more accurate in later iterations. In contrast, the prediction errors of *[Tomanek09]* remain in the corpus and cannot be corrected. As a result, the error curve of *[ReEst(Train)]* sharply decreases, which is similar to the curve of *[Tomanek09]* but continues decreasing to lower error level.

Figure 6.10 shows the F_1 among the system without any re-estimation (*[Tomanek09]*), the system with the re-estimation in the training (*[ReEst(Train)]*), and the system with the re-estimation in both the training and the annotation phases (*[ReEst(Annotation,Train)]*). Although the F_1 of *[ReEst(Train)]* reached a higher level than *[Tomanek09]*, it requires more labeled tokens. However, after we combine the re-estimation in the annotation phase, we can reduce the annotation cost but keep the high F_1 level as stated in Section 6.4.

6.6 Token Sampling

We compare between the 2 sampling strategies, the sequence sampling and the token sampling. In the token sampling, 500 sequences with the most informative tokens are

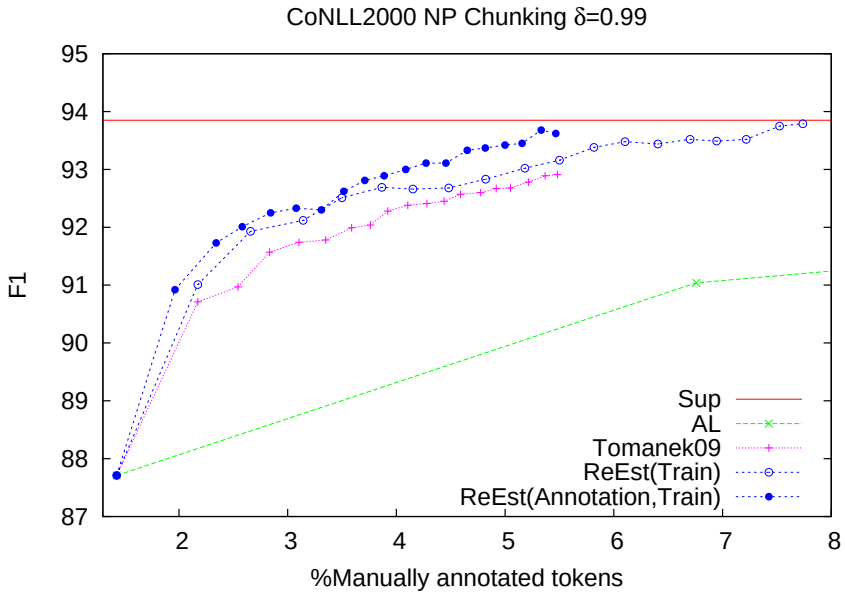


Figure 6.10: F_1 of the model with and without the re-estimation in the training phase

sampled. Only the most informative tokens in each selected sequence will be annotated in one iteration. In the sequence sampling, 500 most informative sequences are sampled. All informative tokens in the selected sequences are annotated. The *[Seq-Sampling]* system is exactly the same system as the *[ReEst(Annotation,Train)]* in Section 6.5. In both of the sampling strategies, high-confidence tokens, which are uninformative, are left for the implicit annotation in the training phase.

Figure 6.11 shows that the token sampling strategy, which provides more opportunity for confidence re-estimation than the sequence sampling, achieves higher F_1, with more labeled token cost. We also compare the token sampling with different confidence threshold. The system with $\delta = 0.90$ also achieves the supervised F_1 with fewer labeled tokens than the system with higher threshold, $\delta = 0.99$.

6.7 The Computational Time

The re-estimation in each process can reduce the annotation cost in terms of labeled tokens, but the re-estimation requires additional computational cost. There are several factors that affect the computational cost. In this thesis, we take 3 factors into account, the corpus size, the re-estimation step, and the number of label candidates. We will

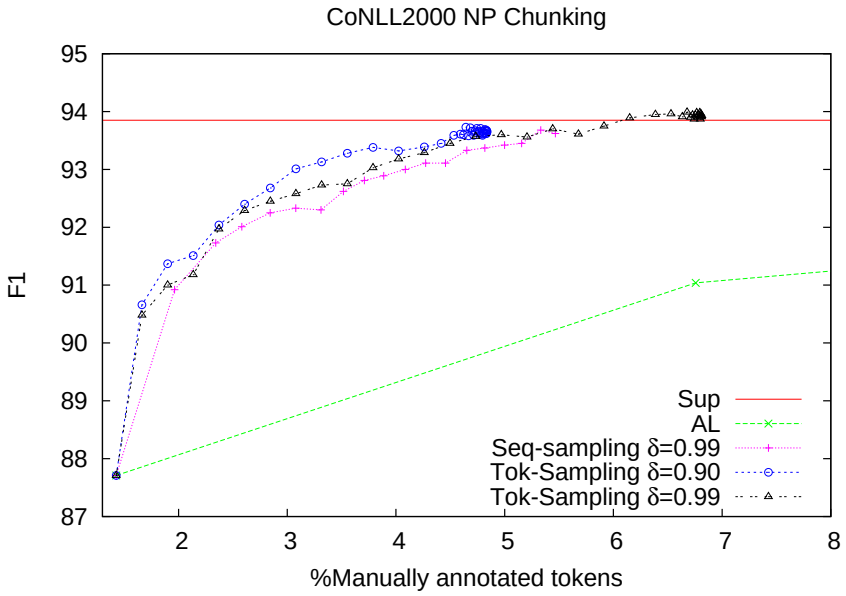


Figure 6.11: F_1 of the sequence sampling and the token sampling

discuss the first two factors in this section. The last factor will be discussed in Section 6.8.

We plot the number of tokens/sequences and the CPU time of *[AL]* using green line and point in Figure 6.12. A larger corpus which contains more sequences, will need more training time than a smaller corpus; since the training will evaluate the current parameters on all training sequences.

Secondly, we compare the computational time of *[AL]* and *[ReEst(Annotation)]* to determine the effect of the re-estimation in the annotation. Although the second setting requires additional time in the re-estimation, the overall CPU time is less than the time in *[AL]*. We argue that the re-estimation in the annotation takes few seconds, but the training time is substantially reduced due to the automatically labeled tokens. There are few differences between the model prediction and the automatically annotated label of the high-confidence tokens. Therefore, the optimization will not take a long time for the updating parameters of these tokens.

Thirdly, we study the impact of the re-estimation in the training phase. From Figure 6.12, *[ReEst(Train)]* requires less training time than *[AL]*. We suggest again that the optimization takes a short time since there are few differences between the predicted and the marginalized labels. Finally, *[ReEst(Annotation,Train)]* requires less

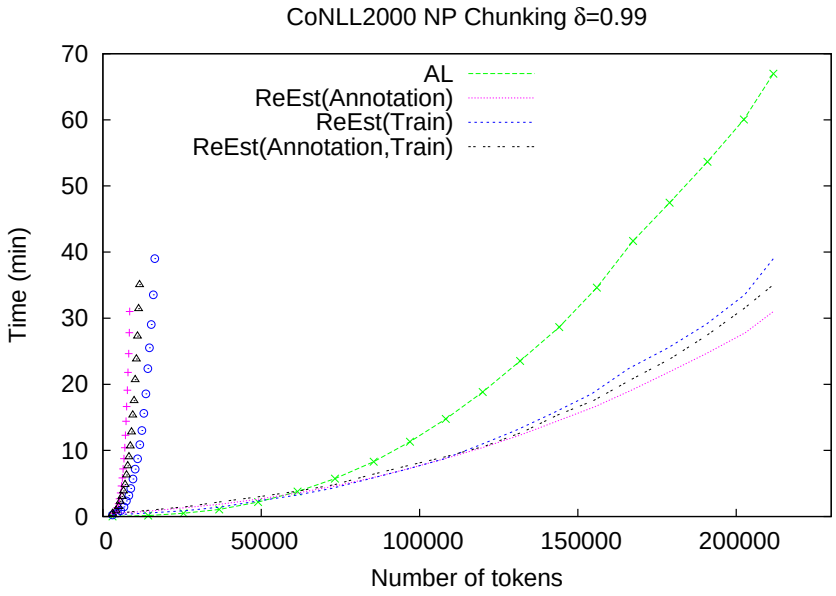


Figure 6.12: CPU time of the training on different corpus size

time than *[ReEst(Train)]* because the re-estimation in the annotation makes the prediction confidence of tokens relatively stable and will not change in later iterations.

6.8 The Simplified Sampling

In this section, we estimate the impact of the number-of-label-candidate factor at the training time, and the simplified sampling strategy which is intended to reduce the training cost due to this factor.

We split the all-phrase labeling task into several single-type labeling tasks. We compare the CPU time of a single iteration of *[AL]* between the two labeling strategies using CoNLL2000 data set in Figure 6.13. The original and simplified data sets have identical training instances. The difference is only the number of target classes, which will affect the feature complexity. The simplified chunking tasks clearly require less computational time.

In the last experiment, we compare the cumulative CPU time of the simplified labeling, and all phrase labeling in Figure 6.14. Although the all-phrase chunking is split to several sub tasks, the cumulative CPU time of the simplified labeling is still comparable to the full labeling.

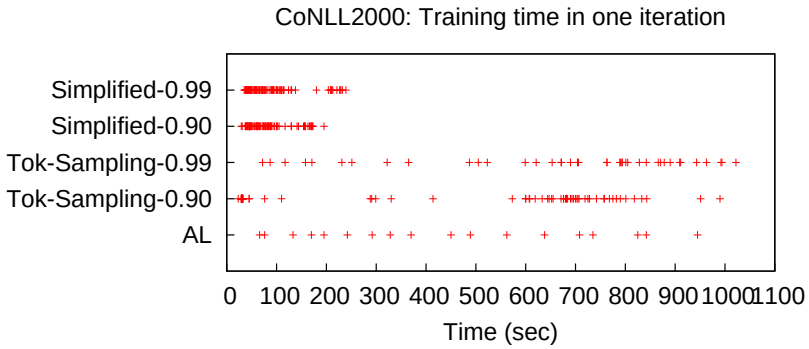


Figure 6.13: Single-iteration CPU time of the all-phrase chunking and single-type chunking tasks

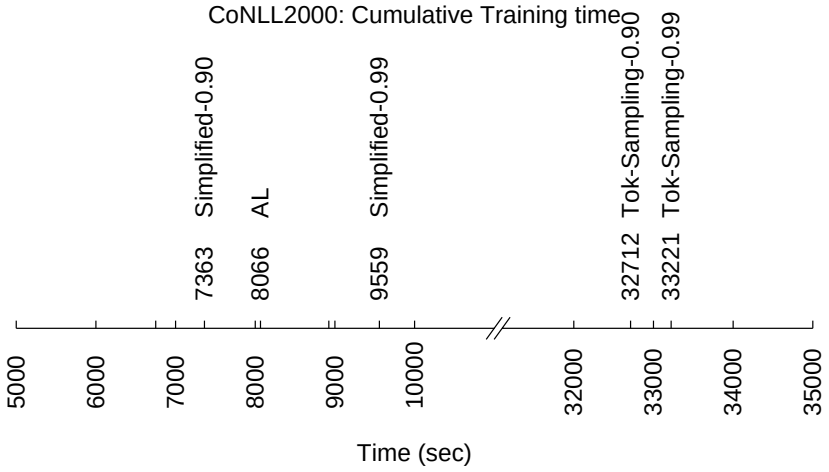


Figure 6.14: Cumulative CPU time of the all-phrase chunking and single-type chunking tasks

Table 6.6: Result of each system on CoNLL2000 and CoNLL2003. Boldface figures indicate that the F_1 is not significantly different from the supervised F_1. Shaded cells show the main disadvantage of the systems.

CoNLL2000

Systems	F_1	Num. Lab. Tokens	Time/Iter	Cum. Time
[AL]	93.42	95.54	480	8066
[Tok-Sampling]-0.90	93.41	5.87	723	32712
[Tok-Sampling]-0.99	93.46	7.09	772	33221
[Simplified]-0.90	92.98	7.67	83	7363
[Simplified]-0.99	93.14	11.68	88	9559

CoNLL2003

Systems	F_1	%Labeled Tokens	Time/Iter	Cum.Time
[AL]	81.49	88.52	1972	42056
[Tok-Sampling]-0.90	81.21	7.23	2790	78118
[Tok-Sampling]-0.99	81.33	9.29	1707	59762
[Simplified]-0.90	80.46	6.35	560	47244
[Simplified]-0.99	81.11	7.21	638	58817

6.9 Final Result

Table 6.6 shows the comparison of the proposed framework to the *[AL]* baseline. The proposed all-type labeling system achieves the supervised F_1 on both data sets at the expense of longer average idle time of an annotator between iterations. On the other hand, the proposed single-type labeling system achieves slightly lower F_1 with much shorter idle time than the baseline system. The cumulative CPU time of the simplified task is also close to the cumulative CPU time of the *[AL]* baseline.

Chapter 7

Conclusion and Future Work

We have proposed an active learning framework for sequence labeling that incorporates the prediction confidence in the annotation, re-training, and sampling phases of active learning. The proposed framework could succeed in reducing the manually labeled tokens from the supervised learning, but still be able to achieve the supervised F_1. The confidence re-estimation could increase the prediction confidence in the annotation, correct prediction errors of early models in the training phase, and increase the reliability of the instance selection in the sampling phase.

The simplified sampling efficiently reduced the training time from the general framework. The simplified sampling achieved the competitive accuracy to the supervised system while it required much less labeled tokens than the supervised one. The proposed sampling also required reasonable training time compared to the exhaustive time of the all-entity labeling framework since the simplified labeling contains less number of target labels.

One can refine this work on the tagger itself since training CRFs is time consuming. A light-weight tagger such as perceptron might be more suitable to active learning than CRFs. Moreover, the annotation cost modeling must be more realistic in order to adjust the appropriate query size parameter. For example, the current query size is set based only on the informativeness score. The query size should also be set to represent the actual time required in the annotation. We can directly incorporate a cost model such as the one in [54], which can represent the actual human annotation time, to our query strategy.

Our framework is not limited to the sequence labeling task but can be extended to the general graph labeling. Since the factor graph is defined over a general graph. We can estimate the marginal probability on a general graph using Belief Propagation [36]. We leave the task to the future work.

References

[1] Naoki Abe and Hiroshi Mamitsuka. **Query Learning Strategies Using Boosting and Bagging**. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 1–9, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. 20

[2] Rie Kubota Ando and Tong Zhang. **A High-performance Semi-supervised Learning Method for Text Chunking**. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 1–9, Morristown, NJ, USA, 2005. Association for Computational Linguistics. 27

[3] Dana Angluin. **Queries and Concept Learning**. *Machine Learning*, **2**(4):319–342, 1998. 17

[4] Jason Baldridge and Miles Osbourne. **Active Learning and the Total Cost of Annotation**. In *Proceedings of Empirical Method in Natural Language Processing*, EMNLP '04, pages 9–16. Association for Computational Linguistics, 2004. 23

[5] Kedar Bellare and Andrew McCallum. **Learning Extractors from Unlabeled Text using Relevant Databases**. In *Proceedings of the Sixth International Workshop on Information Integration on the Web*, IIWEB-07. Association for the Advancement of Artificial Intelligence, 2007. 14

[6] Kedar Bellare and Andrew McCallum. **Generalized Expectation Criteria for Bootstrapping Extractors using Record-text Alignment**. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 131–140, Morristown, NJ, USA, 2009. Association for Computational Linguistics. 29

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, chapter 8. Graphical Models. Springer-Verlag, 2006. 9

[8] Avrim Blum and Tom Mitchell. **Combining Labeled and Unlabeled Data with Co-training**. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. Association for Computing Machinery. 27

[9] Klaus Brinker. **Active Learning of Label Ranking Functions**. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 129–136, New York, NY, USA, 2004. ACM. 19

[10] Colin Campbell, Nello Cristianini, and Alex J. Smola. **Query Learning with Large Margin Classifiers**. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 111–118, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 19

[11] Jean Carletta. **Assessing Agreement on Classification Tasks: the Kappa Statistic**. *Computational Linguistics*, **22**(2):249–254, 1996. 40

[12] Michele Clamp, Ben Fry, Mike Kamal, Xiaohui Xie, James Cuff, Michael F. Lin, Manolis Kellis, Kerstin Lindblad-Toh, , and Eric S. Lander. **Distinguishing Protein-coding and Noncoding Genes in the Human Genome**. *Proceedings of the National Academy of Sciences of the United States of America*, **104**(49):19428–19433, December 2007. 1

[13] Stephen Clark, James R. Curran, and Miles Osborne. **Bootstrapping POS Taggers using Unlabelled Data**. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, **4** of *CONLL '03*, pages 49–55, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. 27

[14] David Cohn, Les Atlas, and Richard Ladner. **Improving Generalization with Active Learning**. *Machine Learning*, **15**(2):201–221, 1994. 6, 17

[15] Aron Culotta and Andrew McCallum. **Reducing Labeling Effort for Structured Prediction Tasks**. In *Proceedings of the 20th National Conference on Artificial Intelligence*, AAAI '05, pages 746–751. AAAI Press, 2005. 19

[16] Ido Dagan and Sean P. Engelson. **Committee-Based Sampling For Training Probabilistic Classifiers**. In *Proceedings of the Twelfth International Conference on Machine Learning*, ICML '95, pages 150–157. Morgan Kaufmann Publishers Inc., 1995. 17, 20

[17] Sajib Dasgupta and Vincent Ng. **Mine the Easy, Classify the Hard: a Semi-supervised Approach to Automatic Sentiment Classification**. 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*, **2** of *ACL-IJCNLP '09*, pages 701–709, Morristown, NJ, USA, August 2009. Association for Computational Linguistics. 28

[18] Gregory Druck, Burr Settles, and Andrew McCallum. **Active Learning by Labeling Features**. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 81–90, Morristown, NJ, USA, 2009. Association for Computational Linguistics. 24

[19] Heidi Fox, Richard Schwartz, Rebecca Stone, Ralph Weischedel, and Walter Gadz. **Learning to Extract and Classify Names from Text**. In *IEEE International Conference on Systems, Man, and Cybernetics*, IEEE SMC '98, pages 1668–1673, San Diego, CA , USA, October 1998. 1

[20] Nir Friedman, Dan Geiger, and Moises Goldszmidt. **Bayesian Network Classifiers**. *Machine Learning*, **29**:131–163, November 1997. 11

[21] L. Gillick and S.J. Cox. **Some Statistical Issues in the Comparison of Speech Recognition Algorithms**. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1989.*, **1** of *ICASSP 1989*, pages 532–535, May 1989. 45

[22] David Heckerman. **A Tutorial on Learning With Bayesian Networks**. Technical Report MSR-TR-95-06, Microsoft Research, 1995. 10

[23] Rong Hu, Brian Mac Namee, and Sarah Jane Delany. **Off to a Good Start: Using Clustering to Select the Initial Training Set in Active Learning**. In *Proceedings of the Twenty-First International Florida Artificial*

Intelligence Research Society Conference, FLAIRS '10, pages 26–31. AAAI Press, 2010. 21

[24] Jaeho Kang, Kwang Ryel Ryu, and Hyuk-Chul Kwon. **Using Cluster-Based Sampling to Select Initial Training Set for Active Learning in Text Classification**. In *Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. 21

[25] Donghyun Kim, Hyunjung Lee, Choong-Nyoung Seon, Harksoo Kim, and Jungyun Seo. **Speakers' Intention Prediction using Statistics of Multi-level Features in a Schedule Management Domain**. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 229–232, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. 1

[26] Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. **MMR-based Active Machine Learning for Bio Named Entity Recognition**. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '06, pages 69–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. 24

[27] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. **Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data**. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 12

[28] Justin Lamb, Emily D. Crawford, David Peck, Joshua W. Modell, Irene C. Blat, Matthew J. Wrobel, Jim Lerner, Jean-Philippe Brunet, Aravind Subramanian, Kenneth N. Ross, Michael Reich, Haley Hieronymus, Guo Wei, Scott A. Armstrong, Stephen J. Haggarty, Paul A. Clemons, Ru Wei1, Steven A. Carr, Eric S. Lander, and Todd R. Golub. **The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease**. *Science*, **313**(5795):1929–1935, 2006. 1

[29] David D. Lewis and William A. Gale. **A Sequential Algorithm for Training Text Classifiers**. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. 17

[30] Mingkun Li and Ishwar K. Sethi. **Confidence-Based Active Learning**. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **28**:1251–1261, August 2006. 22

[31] D. C. Liu and J. Nocedal. **On the Limited Memory BFGS Method for Large Scale Optimization**. *Mathematical Programming: Series A and B*, **45**(3):503–528, 1989. 13

[32] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. **Building a Large Annotated Corpus of English: the Penn Treebank**. *Computational Linguistics*, **19**:313–330, June 1993. 3

[33] Andrew McCallum and Kamal Nigam. **Employing EM and Pool-Based Active Learning for Text Classification**. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 350–358, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. 20, 21

[34] Graham Neubig and Shinsuke Mori. **Word-based Partial Annotation for Efficient Corpus Construction**. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, LREC'10, pages 2723–2727, Valletta, Malta, May 2010. European Language Resources Association (ELRA). 3, 31, 49

[35] David Opitz and Richard Maclin. **Popular Ensemble Methods: An Empirical Study**. *Journal of Artificial Intelligence Research*, **11**:169–198, 1999. 27

[36] Judea Pearl. **Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach**. In *Proceedings of the Second National Conference on Artificial Intelligence. AAAI-82:*, pages 133–136, Menlo Park, California, 1982. American Association for Artificial Intelligence, AAAI Press. 62

[37] John C. Platt. **Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods**. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999. 19

[38] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. **Self-taught Learning: Transfer Learning from Unlabeled Data**. In *Proceedings of the Twenty-fourth International Conference on Machine Learning*, ICML '07, pages 759–766, New York, NY, USA, 2007. Association for Computing Machinery. 27

[39] Lance A. Ramshaw and Mitchell P. Marcus. **Text Chunking using Transformation-Based Learning**. In *Proceedings of the Third Workshop on Very Large Corpora*, WVLC-3, pages 82–94. Association for Computational Linguistics, 1995. 43

[40] Adwait Ratnaparkhi. **A Maximum Entropy Model for Part-Of-Speech Tagging**. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '96, pages 133–142, Morristown, NJ, USA, 1996. Association for Computational Linguistics. 17

[41] Dan Roth and Kevin Small. **Margin-Based Active Learning for Structured Output Spaces**. In *Proceedings of the 17th European Conference on Machine Learning*, ECML' 06, pages 413–424, 2006. 19, 23

[42] Manabu Sassano and Sadao Kurohashi. **Using Smaller Constituents Rather than Sentences in Active Learning for Japanese Dependency Parsing**. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 356–365, Uppsala, Sweden, July 2010. Association for Computational Linguistics. 24, 28

[43] Greg Schohn and David Cohn. **Less is More: Active Learning with Support Vector Machines**. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 839–846, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 19, 22

[44] Burr Settles and Mark Craven. **An Analysis of Active Learning Strategies for Sequence Labeling Tasks**. In *Proceedings of the 2008 Conference on*

Empirical Methods in Natural Language Processing, EMNLP '08, pages 1070–1079, Morristown, NJ, USA, 2008. Association for Computational Linguistics. 6, 19, 21, 23, 28

[45] H. Sebastian Seung, M. Opper, and Haim Sompolinsky. **Query by Committee**. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA, 1992. Association for Computing Machinery. 20

[46] Claude Shannon, Noshirwan Petigara, and Satwiksai Seshasai. **A Mathematical Theory of Communication**. *Bell System Technical Journal*, **27**:379–423, 1948. 19

[47] Kathrin Spreyer and Jonas Kuhn. **Data-driven Dependency Parsing of New Languages using Incomplete and Noisy Training Data**. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 12–20, Morristown, NJ, USA, 2009. Association for Computational Linguistics. 28

[48] David G. Stork and Chuck P. Lam. **Open Mind Animals: Insuring the Quality of Data Openly Contributed over the World Wide Web**. Technical Report WS-00-05, American Association for Artificial Intelligence, 2000. 17

[49] Jun Suzuki and Hideki Isozaki. **Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data**. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, ACL '08, pages 665–673, Columbus, Ohio, June 2008. Association for Computational Linguistics. 27, 42

[50] Erik F. Tjong Kim Sang and Sabine Buchholz. **Introduction to the CoNLL-2000 Shared Task: Chunking**. In *Proceedings of the Fourth conference on Computational natural language learning and the Second workshop on Learning language in logic*, CoNLL-2000, pages 127–132, Morristown, NJ, USA, 2000. Association for Computational Linguistics. 41, 44

[51] Erik F. Tjong Kim Sang and Fien De Meulder. **Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity**

Recognition. In *Proceedings of the Seventh Conference on Computational Natural Language Learning*, CoNLL-2003, 2003. 42

[52] Erik F. Tjong Kim Sang and Jorn Veenstra. **Representing Text Chunks**. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '99, pages 173–179. Association for Computational Linguistics, 1999. 41

[53] Katrin Tomanek and Udo Hahn. **Semi-Supervised Active Learning for Sequence Labeling**. 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*, ACL-IJCNLP '09, pages 1039–1047. Association for Computational Linguistics, August 2009. 6, 19, 23, 24, 28, 29, 30, 49

[54] Katrin Tomanek, Udo Hahn, Steffen Lohmann, and Jürgen Ziegler. **A Cognitive Cost Model of Annotations Based on Eye-Tracking Data**. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '08, pages 1158–1167, Uppsala, Sweden, July 2010. Association for Computational Linguistics. 61

[55] Simon Tong and Daphne Koller. **Support Vector Machine Active Learning with Applications to Text Classification**. *Journal of Machine Learning Research*, **2**:45–66, March 2002. 19

[56] Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. **Training Conditional Random Fields using Incomplete Annotations**. In *Proceedings of the 22nd International Conference on Computational Linguistics*, COLING '08, pages 897–904, Morristown, NJ, USA, 2008. Association for Computational Linguistics. 14, 24, 28, 30, 36

[57] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. **Accelerating the Annotation of Sparse Named Entities by Dynamic Sentence Selection**. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, BioNLP '08, pages 30–37, Morristown, NJ, USA, 2008. Association for Computational Linguistics. 28

[58] S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. **Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods**. In *Proceedings of the Twenty-third International Conference on Machine Learning*, ICML '06, pages 969–976, New York, NY, USA, 2006. Association for Computer Machinery. 13

[59] Andreas Vlachos. **A Stopping Criterion for Active Learning**. *Computer Speech and Language*, **22**:295–312, July 2008. 22

[60] Luis von Ahn and Laura Dabbish. **Labeling Images with a Computer Game**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM Press. 17

[61] Dan Wu, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. **Domain Adaptive Bootstrapping for Named Entity Recognition**. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP '09, pages 1523–1532, Morristown, NJ, USA, 2009. Association for Computational Linguistics. 29

[62] Rong Yan, Jie Yang, and Alexander Hauptmann. **Automatically Labeling Video Data using Multi-class Active Learning**. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, **1** of *ICCV '03*, pages 516–523, October 2003. 19

[63] David Yarowsky. **Unsupervised Word Sense Disambiguation Rivaling Supervised Methods**. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Morristown, NJ, USA, 1995. Association for Computational Linguistics. 27

[64] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. **Active Learning with Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification**. In *Proceedings of the 22nd International Conference on Computational Linguistics*, **1** of *COLING '08*, pages 1137–1144, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. 21