Voted Co-training for Bootstrapping Sense Classifiers

Rada MIHALCEA[1]

Abstract. This paper introduces voted co-training, a bootstrapping method that combines co-training with majority voting, with the effect of smoothing the learning curves, and improving the average performance. Voted co-training was evaluated on a word sense classification problem, with significant improvements observed over basic co-training algorithms. Various empirical parameter selection methods for co-training are investigated, with various degrees of error reduction.

1 Introduction

The task of word sense disambiguation consists of assigning the most appropriate meaning to a polysemous word within a given context. Most of the efforts in solving this problem were concentrated so far towards *supervised* learning, where each sense tagged occurrence of a particular word is transformed into a feature vector, which is then used in an automatic learning process. While these algorithms usually achieve the best performance, as compared to their unsupervised or knowledge-based alternatives, their applicability is however limited to those words for which sense tagged data is available, and their accuracy is strongly connected to the amount of labeled data available at hand.

In this paper, we investigate methods for building sense classifiers when only relatively few annotated examples are available. We explore bootstrapping methods using simple co-training, and co-training improved with majority voting (*voted co-training*), and evaluate their performance for building sense classifiers. We also show that classifiers built for different words have different behavior during the bootstrapping process. Given the set of parameters that can influence the performance of a co-training process (growth size, pool size, and number of iterations, as detailed later), we experiment with various parameter settings, using per-word and global parameter settings, and show that the highest overall error rate reduction is achieved with a global parameter scheme using voted co-training.

The paper is organized as follows. We first overview the general approach of co-training for natural language learning. We then introduce the problem of supervised word sense disambiguation, and define local and topical basic classifiers. We investigate the applicability of co-training to supervised word sense disambiguation, starting with these basic classifiers, and perform comparative evaluations of basic co-training and voted co-training, for various empirical parameter settings.

2 Co-training for Natural Language Learning

Co-training [2] is a bootstrapping method that aims to improve the performance of a supervised learning algorithm by incorporating large amounts of unlabeled data into the training data set. Shortly, co-training algorithms work by generating several classifiers trained on the input labeled data, which are then used to tag new unlabeled data. From this newly annotated data, the most confident predictions are sought, which are subsequently added to the set of labeled data. The process may continue for several iterations.

Figure 1 illustrates the general co-training process. Starting with a set of labeled and unlabeled data, the bootstrapping algorithm aims to improve the classification performance, by integrating examples from the unlabeled data into the labeled data set. At each iteration, the class distribution in the labeled data is maintained, by keeping a constant ratio across classes between already labeled examples and newly added examples; the role of this step is to avoid introducing imbalance in the training data set. For co-training, the algorithm requires two different views (two different classifiers C_1 and C_2) that interact in the bootstrapping process.

In natural language learning, co-training was applied to statistical parsing [15], reference resolution [10], [11], part of speech tagging [5], statistical machine translation [4], and others, and was generally found to bring improvement over the case when no additional unlabeled data are used.

One important aspect of co-training consists of the relation between the views used in learning. In the original definition of co-training, Blum and Mitchell [2] state *conditional independence of the views* as a required criterion for co-training to work. In recent work, Abney [1] shows that the independence assumption can be relaxed, and co-training is still effective under a weaker independence assumption. He is proposing a greedy algorithm to maximize agreement on unlabelled data, which produces good results in a co-training experiment for named entity classification. Moreover, Clark et al. [5] show that a *naive* co-training process that does not explicitly seek to maximize agreement on unlabelled data can lead to similar performance, at a much lower computational cost. In this work, we apply co-training by identifying two different feature sets based on a "local versus topical" feature split, which represent potentially independent *views* for word sense classification, as shown in Section 4.

Co-training parameters

Three different parameters can be set in the co-training process, and usually the performance achieved through bootstrapping depends on the value chosen for these parameters.

- *Iterations (I)* Number of iterations.
- *Pool size (P)* Number of examples selected from the unlabeled set U for annotation at each iteration.
- *Growth size (G)* Number of most confidently labeled examples that are added at each iteration to the set of labeled data L.

As previously noticed [12], there is no principled method for selecting optimal values for these parameters, which is an important disadvantage of these algorithms.

[1] Department of Computer Science and Engineering, University of North Texas, rada@cs.unt.edu

Given:
- *A set L of labeled training examples*
- *A set U of unlabeled examples*
- *Classifiers C_i*

1. Create a pool U' of examples by choosing P random examples from U.
2. Loop for I iterations:
2.1 Use L to individually train the classifiers C_i, and label the examples in U'
2.2 For each classifier C_i select G most confidently labeled examples and add them to L, while maintaining the class distribution in L.
2.3 Refill U' with examples from U, to keep U' at a constant size of P examples

Figure 1. General co-training process using labeled and unlabeled data

Feat.	Description
CW (L)	The word AW itself
CP (L)	The part of speech of the word AW
CF (L)	Word forms and their part of speech for a window of K words surrounding AW
COL (L)	Collocations formed with maximum K words surrounding AW
HNP (L)	The head of the noun phrase to which AW belongs, if any
SK (T)	Maximum of M keywords occurring at least N times are determined for each sense of the ambiguous word. The value of this feature is either 0 or 1, depending if the current example contains one of the determined keywords or not.
VB (L)	The first verb before AW.
VA (L)	The first verb after AW.
NB (L)	The first noun before AW.
NA (L)	The first noun after AW.
VO (L)	Verb-object relation involving AW
SV (L)	Subject-verb relation involving AW

Table 1. Commonly used features for word sense disambiguation. AW denotes the current (ambiguous) word. Feature type is indicated as local (L) or topical (T).

In the following, we describe the general framework of supervised word sense disambiguation, and introduce several basic sense classifiers that are used in co-training experiments. Next, we explore various algorithms for empirical selection of the three co-training parameters, and compare results obtained with basic co-training, and co-training improved with majority voting.

3 Supervised Word Sense Disambiguation

Supervised word sense disambiguation systems work under the assumption that several annotated examples are available for a target ambiguous word. These examples are used to build a classifier that automatically learns clues useful for the disambiguation of the given polysemous word, and then applies these clues to the classification of new unlabeled instances.

First, the examples are pre-processed and annotated with morphological or syntactic tags. Next, each sense-tagged example is transformed into a feature vector, suitable for an automatic learning process. There are two main decisions that one takes in the construction of such a classifier: (1) What features to extract from the examples provided, to best model the behavior of the given ambiguous word; (2) What learning algorithm to use for best performance.

Preprocessing

During preprocessing, SGML tags are eliminated, the text is tokenized, and part of speech tags are assigned using Brill tagger [3]. Collocations are identified using a sliding window approach, where a collocation is considered to be a sequence of words that forms a compound concept defined in WordNet. During this process, all collocations that include the target word are identified, and the examples that use a collocation are removed from the training/test data. For instance, examples referring to *short circuit* are removed from the data set for *circuit*, so that a separate learning process is performed for each lexical unit.

Features that are good indicators of word sense

Previous work on word sense disambiguation has acknowledged several local and topical features as good indicators of word sense. These include surrounding words and their part of speech tags, collocations, keywords in contexts. More recently, other possible features have been investigated: bigrams in context, named entities, syntactic features, semantic relations with other words in context. Table 1 lists commonly used features in word sense disambiguation (list drawn from a larger set of features compiled by [8]).

Supervised learning for word sense disambiguation

Related work in supervised word sense disambiguations includes experiments with a variety of learning algorithms, with varying degrees of success: Bayesian learning, decision trees, decision lists, memory based learning, and others. An experimental comparison of seven learning algorithms used to disambiguate the meaning of the word *line* is presented in [9].

Basic Classifiers for Word Sense Disambiguation

Several basic word sense disambiguation classifiers can be implemented using feature combinations from Table 1, and feature vectors can be plugged into any learning algorithm. We use Naive Bayes, since it was previously shown that in combination with the features we consider, can lead to a state-of-the-art disambiguation system [7]. Moreover, Naive Bayes is particularly suited for co-training and self-training, since it provides confidence scores and is efficient in terms of training and testing time. The two separate views required for co-training are defined using a local versus topical feature split.

A local classifier: A local classifier is implemented using all local features listed in Table 1 (features marked with (L) in Table 1).

A topical classifier: The topical classifier relies on features extracted from a large context (features marked with (T) in Table 1). We use the SK feature, and extract at most ten keywords for each word sense, each occurring for at least three times in the annotated corpus.

A global classifier: Finally, the global classifier integrates all local and topical features, again using a Naive Bayes classifier.

4 Co-training for Word Sense Disambiguation

We investigate the application of co-training for bootstrapping sense classifiers, and explore methods for selecting values for the bootstrapping parameters.

The data set used in this study consists of training and test data made available during the English lexical sample task in the SENSEVAL-2 evaluation exercise (http://www.senseval.org). In addition to these data sets, a large raw corpus of unlabeled examples

is constructed for each word, with text snippets consisting of three consecutive sentences extracted from the British National Corpus. Given the large number of runs performed for each word (2,120 runs per word, for various parameter settings), the experiments focus on nouns only. Similar observations are however expected to hold for other parts of speech. Table 2 lists all the words used in the experiments. For each word, it lists: size of training[2], test, raw data[3]; precision of the basic classifier (the global classifier is used as a baseline); precision during bootstrapping experiments, for various parameter settings.

To run the co-training experiments, we use the local and topical classifiers described in Section 3, which represent two different views for this problem, generated by a "local versus topical" feature split.

Unlike previous applications of co-training to natural language learning, where one general classifier is built to cover the entire problem space, supervised word sense disambiguation implies a different classifier for each individual word, resulting eventually in thousands of different classifiers, each with its own characteristics (learning rate, sensitivity to new examples, etc.). Given this large heterogeneous space of classifiers, the co-training process itself may have a heterogeneous behavior, and the selection of best co-training parameters becomes a critical problem.

To address this issue, a range of experiments is performed. First, we explore algorithms to select the bootstrapping parameters, using a validation set: (1) Best global parameter setting (overall and individual); (2) Best per-word parameter selection. We also introduce and evaluate an improved co-training method consisting of a bootstrapping scheme combined with majority voting across several bootstrapping iterations.

The parameter selection methods described in this section make use of the data collected during several co-training runs, for different parameter settings. Evaluations are performed on a validation set consisting of about 20% of the training data – which was set apart for this purpose. For each run, information is collected about: initial size of labeled data set, growth size, pool size, iteration number, precision of basic classifier, precision of boosted classifier.

For the growth size G, a value is chosen from the set: {1, 10, 20, 30, 40, 50, 100, 150, 200}. The pool size P takes one of these values: {1, 100, 500, 1000, 1500, 2000, 5000}. For each setting, 40 iterations are performed. This results in an average of 2,120 classification runs per word. At each run, a pool of P raw examples is annotated, and G most confidently labeled examples are added to the training set from the previous iteration. The performance of the classifier using the augmented training set is evaluated on the validation set, and the precision is recorded. With this range of values for growth size, pool size, and number of iterations, about 60,000 such records are collected for all co-training runs.

[2] The numbers listed in Table 2 for training/test data size and basic classifier precision refer to data sets obtained after removing examples with collocations that include the target word. This explains why the numbers do not always match figures previously reported in SENSEVAL-2 literature. If collocations are added back to the data sets, the precision of the basic classifier is measured at 60.2% – comparable to figures obtained by other systems participating in SENSEVAL-2

[3] The raw corpus for each word is formed with all examples retrieved from the British National Corpus. While this ensures a natural distribution for each word (in terms of number of examples occurring in a balanced corpus), it also leads to discrepancies in terms of raw data size. For words with less than 5000 raw examples, the pool size recorded in the "settings" column represents a round-up to the nearest number from the set of allowed pool values.

4.1 Best global parameter setting

One simple method to select a parameter setting is to determine a global setting that leads to the highest overall boost in precision. Starting with the information collected for the 60,000 runs, for each possible parameter setting, the total relative growth in performance is determined by adding up the relative improvements for all the runs for that particular setting. The best global setting identified in this way is growth size of 50, pool size of 5000, iteration 2. The predictive accuracy of the classifiers obtained for these settings is listed in Table 2 under the *global settings* column. On average, using this scheme for parameter selection, co-training leads to a precision of 55.67%, which translates into a 4.1% error reduction with respect to the 53.84% average precision of the basic classifier.

In a similar approach, the value for each parameter is determined independent of the other parameters. Instead of selecting the best value for all parameters at once, values are selected individually. Again, for each possible parameter value, the total relative growth in performance is determined, and the value leading to the highest growth is selected. Interestingly, the best values identified in this way for the three parameters are growth size of 1, pool size of 1, iteration 1 (i.e. the best classifier is the one "closest" to the basic classifier). The average results are however worse than the baseline (53.49%).

4.2 Best per-word parameter setting

In a second experiment, best parameter settings are identified separately for each word. The setting yielding maximum precision on the validation set is selected as the best setting for a given word, and evaluated on the test data. If multiple settings are identified as leading to maximum precision, settings are prioritized based on (in this order): smallest growth size; largest pool size; number of iterations. Results are listed in Table 2 under the *per-word settings* column, together with the setting identified as optimal on the validation set. There are several words for which significant improvement is observed over the baseline. However, on the average, the performance of the boosted classifiers is worse than the baseline.

5 Voted Co-training

There is a common trend observed for learning curves for co-training, consisting in an increase in performance, followed by a decline. Different classifiers exhibit however a different point of raise or decline, depending on the number of iterations. For instance, the classifier for *circuit* achieves its highest peak at iteration 10 (across all 2,120 co-training runs for different parameter settings), while the classifier for *nation* has the highest boost at iteration 21 – where the performance for *circuit* is already below the baseline. Given this heterogeneous behavior, it is difficult to identify a point of maximum for each classifier, or at least a point where the performance is not below the baseline. Ideally, we would like the learning curves to have a more stable behavior – without sharp raises or drops in precision, and with larger intervals with constant performance, so that the chance of selecting a good number of iterations for each classifier is increased.

We introduce a new bootstrapping scheme that combines co-training with majority voting. During the bootstrapping process, the classifier at each iteration is replaced with a majority voting scheme applied to all classifiers constructed at previous iterations. This change has the effect of "smoothing" the learning curves: it slows down the learning rate, but also yields a larger interval with constant high performance. Although voted co-training and basic co-training face similar difficulties in terms of identifying the highest

Word	Size			Basic	Global setting		Per-word setting		
	train	test	raw	classifier	basic	voted	setting	basic	voted
art	123	52	8012	48.07%	**53.85%**	**53.85%**	50/1500/32	44.23%	**53.85%**
authority	157	80	11034	50.00%	51.25%	57.50%	150/2000/2	57.50%	**58.75%**
bar	205	124	5526	31.45%	31.45%	31.45%	10/1000/5	29.83%	**35.48%**
bum	79	43	361	37.20%	39.53%	44.18%	1/1/40	**46.51%**	44.18%
chair	121	63	5889	**80.95%**	77.78%	79.36%	1/1500/16	77.77%	**80.95%**
channel	78	44	1744	43.18%	34.09%	43.18%	1/2000/2	43.18%	**45.45%**
child	117	60	14192	63.33%	50.00%	51.66%	10/1500/1	55.00%	**65.00%**
church	81	36	7775	52.77%	55.56%	**58.33%**	1/100/23	47.22%	**58.33%**
circuit	108	57	1891	40.35%	45.61%	**49.12%**	100/100/3	31.57%	42.10%
day	245	123	50883	45.52%	52.03%	53.65%	150/1500/3	50.43%	**55.28%**
detention	46	24	638	79.16%	**83.33%**	**83.33%**	-	79.16%	79.16%
dyke	52	26	116	38.61%	**50.00%**	46.15%	1/2000/19	34.61%	38.46%
facility	110	55	1959	67.27%	**69.09%**	**69.09%**	150/1500/13	58.18%	58.18%
fatigue	69	42	437	**73.80%**	71.43%	71.43%	1/1000/1	71.43%	71.43%
feeling	100	51	11214	39.21%	39.21%	**50.98%**	10/500/5	**50.98%**	35.29%
grip	72	39	1718	53.46%	43.59%	48.71%	20/100/20	41.02%	**60.00%**
hearth	60	29	334	44.82%	**48.28%**	44.82%	1/1000/2	41.37%	44.82%
holiday	55	26	5604	**84.61%**	**84.61%**	**84.61%**	-	**84.61%**	**84.61%**
lady	75	39	4677	61.53%	**74.36%**	76.92%	150/2000/10	33.07%	66.66%
material	120	59	10663	37.28%	45.76%	42.37%	100/2000/6	45.76%	**49.15%**
mouth	109	56	8044	50.00%	51.79%	**57.14%**	1/1/40	53.57%	50.00%
nation	60	26	4073	65.38%	69.23%	69.23%	100/2000/23	69.23%	**73.07%**
nature	70	38	14218	39.47%	**50.00%**	47.36%	10/100/3	44.73%	47.36%
post	105	58	10611	37.93%	44.82%	**48.27%**	1/500/28	44.82%	41.37%
restraint	87	43	881	**60.46%**	58.14%	**60.46%**	1/1000/2	53.48%	**60.46%**
sense	83	36	19048	50.00%	47.22%	**58.34%**	150/500/40	25.00%	33.33%
spade	48	28	235	**78.57%**	75.00%	**78.57%**	-	**78.57%**	**78.57%**
stress	77	38	3549	36.84%	52.63%	**55.26%**	1/1500/7	47.36%	52.63%
yew	50	20	167	70.00%	65.00%	**75.00%**	-	70.00%	70.00%
AVERAGE	95	48	7085	53.84%	55.67%	**58.35%**	-	51.73%	56.68%

Table 2. Accuracy of sense classifiers for basic and voted co-training, with global and per-word parameter settings (the growth size / pool size / number of iterations are also indicated)

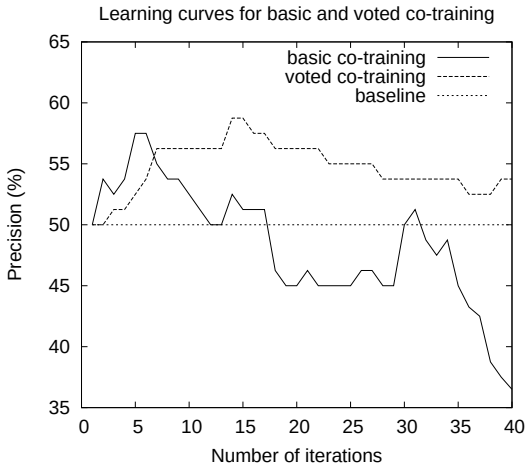

Figure 2. Learning curves for the classifier for the noun *authority*: baseline, simple co-training, and co-training smoothed with majority voting.

peak on the learning curve, the chance of finding a point where the accuracy is significantly higher than the baseline is much increased for voted co-training, since this learning scheme exhibits a larger interval with such higher-than-baseline performance (see e.g. Figure 2).

The main advantage of voted co-training over basic co-training is that it does not allow for "drastical" changes in the learning curve, by combining classifiers built at various iterations through majority voting, and therefore allowing the classifier obtained at a certain iteration to have an impact on the classification process until later stages of bootstrapping. Moreover, the majority voting scheme brings together the strengths of several classifiers, with the effect of improving the overall performance.

Notice that in voted co-training, majority voting is applied on classifiers consisting of iterations of the co-training process itself, and therefore voting is applied on bootstrapped classifiers *across* co-training iterations, with the effect of improving the performance of basic co-training. This is fundamentally different from the approach proposed in [12], where they also apply majority voting in a bootstrapping framework, but in a different setting. They use a majority voting scheme applied to classifiers built on subsets of the labeled data (bagging) to induce several views for the co-training process. In their approach, majority voting is used *at each* co-training iteration to *enable* co-training by predicting labels on unlabeled data.

To some extent, smoothed co-training is related to boosting [6], since both algorithms rely on a growing ensemble of classifiers trained on resamples of the data. However, boosting assumes labeled data and is error-driven, whereas smoothed co-training combines both labeled and unlabeled data and is confidence-driven.

Figure 2 shows the learning curves for simple co-training, and co-training "smoothed" with majority voting, for the word *authority* (for a growth size of 1 and pool size of 1). Notice that the trend for the smoothed curve is still the same – a raise, followed by a decline –

but at a significantly lower pace. With voted co-training, any number of iterations selected in the interval 5-40 still leads to significant improvement over the baseline, unlike the simple unsmoothed curve, where only iterations in the range 3-10 bring improvement over the baseline (followed by two other iterations at random intervals).

The methods for global parameter settings and per-word parameter settings are evaluated again, this time using voted co-training. Table 2 lists the results obtained with basic and voted co-training for the same global/per-word setting. Since the majority voting scheme requires an odd number of classifiers, the number of iterations is rounded up to the next even number (the first iteration is iteration 0, representing the basic classifier, which is also considered during voting).

6 Discussion

In empirical settings, one unique set of parameters for all classifiers seems to perform better than an individual set of parameters customized to each word. For parameter selection using global settings, co-training brings an error reduction of 4.1% over the basic classifier.

As previously noticed [13], it is hard to identify conditionally independent views for real-data problems. Even though we use a "local versus topical" feature split, which divides the features into two separate views on sense classification, there might be some natural dependencies between the features, since they are extracted from the same context, which may weaken the independence condition. However, as theoretically shown in [1], and then empirically in [5], co-training still works under a weaker independence assumption, and the results we obtain concur with these previous observations.

The bootstrapping scheme is improved even more when coupled with majority voting across various iterations. Overall, the highest error reduction is achieved with smoothed co-training using global parameter settings, where an average error reduction of 9.8% is observed with respect to the basic classifier. In terms of efficiency, voted co-training has a running time similar to the basic co-training process, since the application of the majority voting scheme is practically instantaneous.

A comparative analysis of words that benefit from basic/voted co-training with global parameter settings, versus words with little or no improvement obtained through bootstrapping reveals several observations:
(1) Words with accurate basic classifiers cannot be improved through co-training, which agrees with previous observations [14]. For instance, no improvement was obtained for *chair*, *holiday*, or *spade*, which have the basic classifier performing above 75%.
(2) Words with high number of senses (e.g. *bar* – 10 senses, *channel* – 7 senses, *grip* – 11 senses) achieve minimal improvements through co-training. This is probably explained by the fact that the classifiers are misled by the large number of classes (senses), and a large number of errors is introduced since the early stages of co-training.

Even though not all words show benefit from co-training, voted co-training with global parameter settings brings a significant overall error reduction of 9.8% with respect to the basic classifier, and also brings an important improvement over the basic co-training scheme.

7 Conclusion

This paper investigated the application of co-training to supervised word sense disambiguation. Several algorithms for empirical parameter selection were investigated: global settings determined as the best set of parameters across all classifiers, and per-word settings, identified separately for each classifier, both using a validation set. An improved co-training method was also introduced, that combines co-training with majority voting, with the effect of smoothing the learning curves, and improving the average performance. The improved co-training algorithm, applied with a global parameter selection scheme, brought a significant error reduction of 9.8% with respect to the basic classifier, which shows that bootstrapping using an improved form of co-training can be successfully employed in practice for building sense classifiers when only limited amount of annotated data is available. Future work will include comparative analyses and evaluations of: bagging combined with majority voting [12], boosting [6], and voted co-training.

Acknowledgments

This work was partially supported by a National Science Foundation grant IIS-0336793.

REFERENCES

[1] S. Abney, 'Bootstrapping', in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, Philadelphia, PA, (July 2002).
[2] A. Blum and T. Mitchell, 'Combining labeled and unlabeled data with co-training', in *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, (June 1998).
[3] E. Brill, 'Transformation-based error driven learning and natural language processing: A case study in part-of-speech tagging', *Computational Linguistics*, **21**(4), 543–566, (December 1995).
[4] C. Callison-Burch, *Co-training for Statistical Machine Translation*, Master's thesis, University of Edinburgh, 2002.
[5] S. Clark, J. R. Curran, and M. Osborne, 'Bootstrapping POS taggers using unlabelled data', in *Proceedings of the 7th Conference on Natural Language Learning (CoNLL 2003)*, Edmonton, Canada, (June 2003).
[6] Y. Freund and R. Schapire, 'Experiments with a new boosting algorithm', in *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*, Bari, Italy, (July 1996).
[7] Y.K. Lee and H.T. Ng, 'An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation', in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, Philadelphia, (June 2002).
[8] R. Mihalcea, 'Instance based learning with automatic feature selection applied to Word Sense Disambiguation', in *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, (August 2002).
[9] R. Mooney, 'Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning', in *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing (EMNLP 1996)*, Philadelphia, (May 1996).
[10] C. Mueller, S. Rapp, and M. Strube, 'Applying co-training to reference resolution', in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, Philadelphia, (July 2002).
[11] H.T. Ng, B. Wang, and Y.S. Chan, 'Exploiting parallel texts for word sense disambiguation: An empirical study', in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, Sapporo, Japan, (July 2003).
[12] V. Ng and C. Cardie, 'Weakly supervised natural language learning without redundant views', in *Human Language Technology/Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, Edmonton, Canada, (May 2003).
[13] K. Nigam and R. Ghani, 'Analyzing the effectiveness and applicability of co-training', in *Proceedings of the Conference on Information and Knowledge Management (CIKM 2000)*, McLean, VA, (November 2000).
[14] D. Pierce and C. Cardie, 'Limitations of co-training for natural language learning from large datasets', in *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, Pittsburgh, PA, (June 2001).
[15] A. Sarkar, 'Applying cotraining methods to statistical parsing', in *Proceedings of the North American Chapter of the Association for Compuatational Linguistics (NAACL 2001)*, Pittsburgh, (June 2001).