International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:1, 2008

Addressing Scalability Issues of Named Entity Recognition Using Multi-Class Support Vector Machines

Mona Soliman Habib

***Abstract*—**This paper explores the scalability issues associated with solving the Named Entity Recognition (NER) problem using Support Vector Machines (SVM) and high-dimensional features. The performance results of a set of experiments conducted using binary and multi-class SVM with increasing training data sizes are examined. The NER domain chosen for these experiments is the biomedical publications domain, especially selected due to its importance and inherent challenges. A simple machine learning approach is used that eliminates prior language knowledge such as part-of-speech or noun phrase tagging thereby allowing for its applicability across languages. No domain-specific knowledge is included. The accuracy measures achieved are comparable to those obtained using more complex approaches, which constitutes a motivation to investigate ways to improve the scalability of multi-class SVM in order to make the solution more practical and useable. Improving training time of multi-class SVM would make support vector machines a more viable and practical machine learning solution for real-world problems with large datasets. An initial prototype results in great improvement of the training time at the expense of memory requirements.

***Keywords*—**Named entity recognition, support vector machines, language independence, bioinformatics.

I. Introduction

NAMED entity recognition (NER) is one of the important tasks in information extraction, which involves the identification and classification of words or sequences of words denoting a concept or entity. Examples of named entities in general text are names of persons, locations, or organizations. Domain-specific named entities are those terms or phrases that denote concepts relevant to one particular domain. For example, protein and gene names are named entities which are of interest to the domain of molecular biology and medicine. The massive growth of textual information available in the literature and on the Web necessitates the automation of identification and management of named entities in text.

The task of identifying named entities in a particular language is often accomplished by incorporating knowledge about the language taxonomy in the method used. In the English language, such knowledge may include capitalization of proper names, known titles, common prefixes or suffixes, part of speech tagging, and/or identification of noun phrases in text. Techniques that rely on language-specific knowledge may not be suitable for porting to other languages. Moreover, the composition of named entities in literature pertaining to specific domains follows different rules in each, which may or may not benefit from those relevant to general NER.

In previous work [11], a simple architecture that eliminates language and domain-specific knowledge from the named entity recognition process is applied to the English biomedical entity recognition task, as a baseline for other languages and domains. The biomedical field NER remains a challenging task due to growing nomenclature, ambiguity in the left boundary of entities caused by descriptive naming, difficulty of manually annotating large sets of training data, strong overlap among different entities, to cite a few of the NER challenges in this domain. The approach used reduces the pre- and post-processing of the textual data to a minimum and capitalizes on SVM's strong generalization ability to classify the named entities. The accuracy measures achieved are comparable to those obtained using more complex techniques, which encourage us to explore ways to improve the scalability of multi-class support vector machines. In this paper, the results of a set of scalability experiments are reported. These experiments use binary and multi-class SVM with a large set of real-world data from the biomedical literature.

In Section II, the theory of binary and multi-class support vector machines is briefly introduced. Section III describes the experiments' design and summarizes the results of a baseline experiment conducted during the previous work [11] in order to assess the feasibility of our language and domain-independent machine learning NER approach using SVM and high-dimensional features. The baseline experiment design reduces pre-processing to feature extraction and eliminates the use of prior language or domain knowledge. The results of the baseline experiment are a motivation to explore ways to address the scalability issues of the All-Together multi-class SVM approach. Improving scalability of multi-class SVM would provide the research community with a practical and powerful machine learning solution for named entity recognition that promotes the use of high-dimensional features in place of more complex labor and time expensive pre- and post-processing tasks, and simplifies the NER process while achieving good accuracy and performance measures. In Section IV, the results of several sets of single-class and multi-class scalability tests using SVM and increasing training data size are reported and their impact on training time is examined. A sample of preliminary results using a prototype multi-class implementation based on SVM-Perf is also presented.

II. Support Vector Machines

The Support Vector Machine (SVM) is a powerful machine learning tool based on firm statistical and mathematical foundations concerning generalization and optimization theory. SVM is based on Vapnik's statistical learning theory [32] and falls at the intersection of kernel methods and maximum margin classifiers. Support vector machines have been successfully applied to many real-world problems such as face detection, intrusion detection, handwriting recognition, information extraction, and others.

Support Vector Machine is an attractive method due to its high generalization capability and its ability to handle high-dimensional input data. Compared to neural networks or decision trees, SVM does not suffer from the local minima problem, it has fewer learning parameters to select, and it produces stable and reproducible results. However, SVM suffers from slow training especially with non-linear kernels and with large input data size. Support vector machines are primarily binary classifiers. Extensions to multi-class problems are most often performed by combining several binary machines in order to produce the final multi-classification results. The more difficult problem of training one SVM to classify all classes uses much more complex optimization algorithms and are much slower to train than binary classifiers.

A. *Binary Support Vector Classification*

Binary classification is the task of classifying the members of a given set of objects into two groups on the basis of whether they have some property or not. Many applications take advantage of binary classification tasks, where the answer to some question is either a yes or no. For example, product quality control, automated medical diagnosis, face detection, intrusion detection, or finding matches to a specific class of objects. The mathematical foundation of Support Vector Machines and the underlying Vapnik-Chervonenkis dimension (VC Dimension) is described in details in the literature covering the statistical learning theory [1, 2, 15, 18, 24, 32] and many other sources.

The main objective of support vector machines is to find the optimal hyperplane separating positive and negative examples by maximizing the margin between the two classes. In mathematical terms, the problem is to find $f(\mathbf{x}) = (\mathbf{w}^T \mathbf{x}_i + b)$ with maximal margin, such that:

$\mathbf{w}^T \mathbf{x}_i + b = 1$ for data points that are *support vectors*

$\mathbf{w}^T \mathbf{x}_i + b > 1$ for other data points

Assuming a linearly separable dataset, the task of learning coefficients **w** and *b* of support vector machine $f(\mathbf{x}) = (\mathbf{w}^T \mathbf{x}_i + b)$ reduces to solving the following constrained optimization problem:

find **w** and *b* that minimize: $\frac{1}{2}\|\mathbf{w}\|^2$

subject to: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i$

The original optimization problem can be rewritten as its equivalent *dual problem* which finds α that maximizes

$$\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to $\sum_{i=1}^{N} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \forall i$

In the non-linearly separable case, the margin maximization technique may be relaxed by a degree of error in the separation. Slack Variables ξi are introduced to represent the error degree for each input data point. The optimization goal in this case is to maximize the margin while minimizing the slack variables, i.e., to find **w** and b that minimize:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i^2$$

subject to: $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$

In order to solve the non-linearly separable case, SVM introduces the use of a mapping function Φ: RM → F to translate the non-linear input space into a higher dimension feature space where the data is linearly separable. The dual formulation of the optimization problem in feature space is to find α that maximizes:

$$\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

subject to $\sum_{i=1}^{N} \alpha_i y_i = 0,$
$0 \leq \alpha_i \leq C, \quad \forall i$

The resulting SVM is of the form:

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}_i) + b = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b$$

B. *Multi-class Support Vector Classification*

For classification problems with multiple classes, different approaches are developed in order to decide whether a given data point belongs to one of the classes or not. The most common approaches are those that combine several binary classifiers and use a voting technique to make the final classification decision. These include: One-Against-All [32], One-Against-One [21], Directed Acyclic Graph (DAG) [26], and Half-against-half method [23]. A more complex approach is one that attempts to build *one* Support Vector Machine that separates *all* classes at the same time. In this section, these multi-class SVM approaches are briefly introduced. Fig. 1 compares the decision boundaries for three classes using a One-Against-All SVM, a One-Against-One SVM, and an All-Together SVM [1]. The interpretation of these decision boundaries will be discussed the training and classification techniques using each approach are defined.

1) One-Against-All Multi-Class SVM

One-Against-All [32] is the earliest and simplest multi-class SVM. For a *k*-class problem, One-Against-All maximizes *k* hyperplanes separating each class from all the rest by constructing *k* binary SVMs. The *i*th SVM is trained with all the samples from the *i*th class against all the samples from the other classes. To classify a sample *x*, *x* is evaluated by all of the *k* SVMs and the label of the class that has the largest value of the decision function is selected.

Since all other classes are considered negative examples during training of each binary classifier, the hyperplane is optimized for one class only. As illustrated in Fig. 1, unclassifiable regions exist when more than one classifier returns a positive classification for an example *x* or when all classifiers evaluate *x* as negative [1].

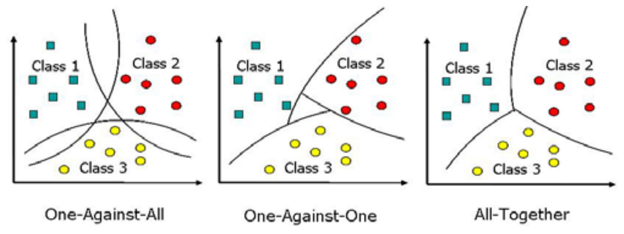

Fig. 1 Comparison of Multi-Class Boundaries

2) One-Against-One or Pairwise SVM

One-Against-One [21] constructs one binary machine between pairs of classes. For a *k*-class problem, it constructs $k(k-1)/2$ binary classifiers. To classify a sample *x*, the sample is evaluated by each of the $k(k-1)/2$ machines. The class that gets the largest value of the decision function by most machines is chosen as the classification of *x*.

Since One-Against-One separates two classes at a time, the separating hyperplanes identified by this approach are tuned better than those found with One-Against-All (Fig. 1). Unclassifiable regions exist only when all classifiers evaluate a sample *x* as negative.

3) Directed Acyclic Graph SVM

Similar to One-Against-One SVM, Directly Acyclic Graph (DAG) [26] trains $k(k-1)/2$ binary classifiers for pairwise classes. To evaluate a sample *x*, this technique builds a DAG ordering the classes 1 through *k* to make a decision. The sample *x* is first evaluated by the first and the last classifier on the DAG and eliminates the lower vote from the DAG. The process is repeated until only one class remains and its label is chosen [12]. Therefore, a decision is reached after (*k* – 1) binary SVM evaluations. Unclassifiable regions are eliminated by excluding one class at a time.

4) Half-Against-Half SVM

Half-Against-Half multi-class SVM [23] is useful for problems where there is a close similarity between groups of classes. Using Half-Against-Half SVM, a binary classifier is built that evaluates one group of classes against another group. The trained model consists of at most $2^{\lceil \log_2 k \rceil}$ binary SVMs. To classify a sample *x*, this technique identifies the group of classes where the sample *x* belongs, than continues to evaluate *x* with a subgroup, and so on, until the final class label is found. The classification process is similar to a decision tree that requires $\lceil \log_2 k \rceil$ evaluations at most.

5) All-Together or All-At-Once SVM

An All-Together multi-classification approach is computationally more expensive yet usually more accurate than all other multi-classification methods. Hsu and Lin [12] note that "as it is computationally more expensive to solve multi-class problems, comparisons of these methods using large-scale problems have not been seriously conducted." The experiments reported in this paper are an attempt to classify a large-scale problem using this approach.

The All-Together approach builds ***one*** SVM that maximizes ***all*** separating hyperplanes at the same time. Training data representing all classes is used to generate the trained model. With this approach, there are no unclassifiable regions as each data point belongs to some class represented in the training dataset. Fig. 1 illustrates the elimination of unclassifiable regions in this case.

The All-together multi-class SVM poses a complex optimization problem as it maximizes all decision functions at the same time [9]. The idea is similar to the One-Against-All approach. It constructs *k* two-class rules where the *m*th function $w_m^T \phi(x) + b$ separates training vectors of the class *m* from the other vectors. There are *k* decision functions but all are obtained by solving one problem. The primal formulation of the optimization problem [1, 12] is to find:

$$\min_{w_m, \xi_i} \frac{1}{2} \sum_{m=1}^{k} w_m^T w_m + C \sum_{i=1}^{l} \xi_i$$

such that,

$$w_{y_i}^T \phi(x_i) - w_m^T \phi(x_i) \geq e_m^i - \xi_i, i = 1,\dots,l$$

$$\text{where } e_i^m \equiv 0, \text{if } y_i = m, \text{and } e_i^m \equiv 1, \text{if } y_i \neq m$$

and the decision function is $\operatorname{argmax}_{m=1,\dots,k} w_m^T \phi(x)$.

Like binary SVM, it is easier to solve the dual problem,

$$\min_{\alpha} f(\alpha) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} K_{i,j} \overline{\alpha}_i^T \alpha_j + \sum_{i=1}^{l} \overline{\alpha}_i^T \overline{e}_i$$

such that, $\sum_{i=1}^{k} \alpha_i^m = 0, i = 1,\dots l, m = 1,\dots,k$

$$\alpha_i^m \leq 0, \text{if } y_i \neq m \text{ and } \alpha_i^m \neq C, \text{if } y_i = m$$

where $K_{i,j} = \phi(x_i)^T \phi(x_j)$

The final decision function is $\operatorname{argmax}_{m=1,\dots,k} \sum_{i=1}^{l} \alpha_i^m K(x_i, x)$.

Algorithms to decompose the problem [12] and to solve the optimization problem [30] have been developed, however, the All-Together multi-class SVM approach remains a daunting task. The training time is very slow which makes the approach so far unusable for real-world problems with a large data set and/or a high number of classes. In this paper, an attempt to use this approach with a large set of real-world data is presented and its scalability issues are examined.

C. *SVM Scalability Challenges*

Bennett and Campbell [5] discuss the common usability and scalability issues of support vector machines. In this section we summarize the SVM scalability challenges noted in the literature and in practice, which include:

- Optimization requires $O(n^3)$ time and $O(n^2)$ memory for single class training, where n is input size (depending on algorithm used). To address this issue, new optimization algorithms continue to be introduced [7, 8, 17, 28].
- Multi-class training time is much higher, especially for All-Together optimization. The experimental results using this approach are reported later in this paper.
- Multi-class performance depends on approach used and deteriorates with more classes.
- Slow training, especially with large input datasets and/or non-linear kernels, which may be addressed by reducing the input data size (pruning [10], chunking [3], clustering [4]), reducing the number of support vectors (model decomposition and shrinking [14]), or reducing feature dimensionality (using a priori clustering [4] or adaptive clustering [6]).

In addition to the scalability issues, tuning support vector machines requires the selection of a suitable kernel function and model parameters. Model parameters are often selected using a grid search, cross-validation, or heuristic-based methods. Selection of a suitable kernel function for the problem at hand is another designer-determined factor. Using grid search and cross-validation to select the best model may not be feasible with large datasets. The scalability and performance experiments presented later show that smaller data sets may be used instead.

III. Baseline Experiments

The baseline experiments [11] aim to identify biomedical named entities using Support Vector Machines (SVM) [32], due to their generalization capability and their ability to handle high-dimensional feature and input space. The training and testing data use the JNLPBA-04 shared task [20] data, which is a subset of the GENIA annotated corpus [19] of MEDLINE articles. The names of proteins, cell lines, cell types, DNA and RNA entities are previously labeled. The named entities are often composed of a sequence of words. The training data includes 2,000 annotated abstracts (consisting of 492,551 tokens). The testing data includes 404 abstracts (consisting of 101,039 tokens) annotated for the same classes of entities. The fraction of positive examples with respect to the total number of tokens in the training set varies from about 0.2% to about 6%. Basic statistics about the data sets as well as the absolute and relative frequencies for named entities within each set can be found in [20].

The systems participating in the JNLPBA-04 task employ a variety of machine learning techniques such as Support Vector Machines (SVM), Hidden Markov Models (HMM), Maximum Entropy Markov Models (MEMM) and Conditional Random Fields (CRF). Five systems adopted SVMs either in isolation [22, 25], or in conjunction with other model [27, 29, 34]. The results of our baseline experiment [11] are compared to the five systems using SVM listed above, in addition to those reported in [10]. TABLE I summarizes the performance comparison results in terms of recall, precision, and $F_{\beta=1}$-score.

The training and test data pre-processing involves morphological and contextual features extraction only. No language-specific pre-processing such as part-of-speech or noun phrases tagging is used. No dictionaries, gazetteers, or other domain-specific knowledge are used.

A. *Features Selection*

The training and testing data is preprocessed using the JFEX software [10] in order to extract morphological and contextual features. The generated feature space is very large, including over a million different features in the complete training dataset. All features are binary, i.e., each feature denotes whether the current token possesses this feature (one) or not (zero).

The morphological features extracted are:

- Capitalization: token begins with a capital letter.
- Numeric: token is a numeric value.
- Punctuation: token is a punctuation.
- Uppercase: token is all in uppercase.
- Lowercase: token is all in lowercase.
- Single character: token length is equal to one.
- Symbol: token is a special character.
- Includes hyphen: one of the characters is a hyphen.
- Includes slash: one of the characters is a slash.
- Letters and Digits: token is alphanumeric.
- Capitals and digits: token contains caps and digits.
- Includes caps: some characters are in uppercase.
- General regular expression summarizing word shape.

In addition to the morphological features, a contextual collocation of tokens active over three positions around the token itself is used in order to provide a moving window of consecutive tokens which describes the context of the token relative to its surrounding.

B. *Baseline Performance Results*

A comparison of the performance of the baseline multi-class experiment [11] to other systems using SVM for biomedical NER is presented in TABLE I. It is important to note that the baseline experiment represents a worse-case scenario for the potential performance of the system as it focuses on assessing the feasibility of the simplified approach only and no tuning is performed. The overall recall measure achieved in this case is 62.43%, with a precision measure of 64.50%, and a final F-score of 63.45%.

The language-independent approach used in this experiment performed very close to [25] and better than [22] which both used SVM as the only learning model. Park et al. [25] used character n-grams, orthographic information, word shapes, gene sequences prior knowledge, word variations, part-of-speech tags, noun phrase tags, and word triggers. Lee et al. [22] used lexical features, character n-grams, and part-of-speech tags in a two-phased model based on SVMs. Rössler [27] adapted a NER-system for German to the biomedical field. The system used character n-grams, orthographic information, gene sequences prior knowledge, and word length as features.

TABLE I
BASELINE MULTI-CLASS EXPERIMENT RESULTS VS. SYSTEMS USING JNLPBA-04 DATA

System	Overall Performance Recall/Precision/F-Score
Zhou [34]	76.0 / 69.4 / 72.6
Giuliano [10]	64.4 / 69.8 / 67.0
Song [29]	67.8 / 64.8 / 66.3
Rössler [27]	67.4 / 61.0 / 64.0
Habib [11]	**62.4 / 64.5 / 63.5**
Park [25]	66.5 / 59.8 / 63.0
Lee [22]	50.8 / 47.6 / 49.1
Baseline [20]	52.6 / 43.6 / 47.7

Song et al. [29] used SVM in combination with Conditional Random Fields (CRF) and included character n-grams, orthographic information, and other lexical features in addition to part-of-speech and noun phrase tagging. Giuliano et al. [10] also incorporated part-of-speech tagging and word features of tokens surrounding each analyzed token in addition to features similar to those used in this experiment. In addition, Giuliano et al. [10] pruned the data instances in order to reduce the dataset size by filtering out frequent words from the corpora because they are less likely to be relevant than rare words.

Zhou and Su [34] developed the system that performed best in the JNLPBA-04 task. Zhou and Su [34] used SVM in conjunction with Hidden Markov Models in a more complex learning method. The systems made use of many language-specific and domain-specific knowledge such as character n-grams, orthographic information, gene sequences, gazetteers, part-of-speech tags, word triggers, abbreviations, and cascaded entities. While this system performed better than the baseline multi-class experiment, its heavy use of language and domain-specific prior knowledge contradicts the promise of the language and domain-independent approach [11].

Different machine configurations were tried during the course of the baseline experiments. Due to the CPU-intensive nature of the classification process, especially for the All-Together multi-class case, the first successful multi-class experiment ran on a Pentium IV quad-processor Linux-based machine and completed in 17 days. The experiment was repeated on a Xeon quad-processor 3.6 GHz machine and completed in 4 days. The promising accuracy measures of the language and domain-independent, yet simplified machine learning approach, are a motivation to explore the scalability issues and investigate ways to improve training time of multi-class SVM. The following section reports the results of a series of scalability experiments using single-class and multi-class SVMs and examines the impact of training data size on the training time using different SVM implementations.

IV. SVM SCALABILITY EXPERIMENTS

In this section, the results of several sets of scalability experiments using single-class and multi-class SVM are examined. These experiments use the same training and test datasets described in Section III. The datasets represent a real-world problem, namely the biomedical named entity recognition, to identify the names of proteins, DNA, RNA, cell lines, and cell types in biomedical abstracts. The approach used promotes language and domain independence by eliminating the use of prior language-specific and domain-specific knowledge. Pre-processing of the training and test datasets is limited to extracting morphological and contextual features describing words in the biomedical abstracts and representing each vector with a high-dimensional binary vector. The input dimensionality of the training data exceeds a million features. The training data is composed of 492,551 examples and the test data includes 101,039 tokens.

The scalability experiments train single-class and multi-class support vector machines using chunks of the training dataset with increasing size. The trained model is then used to classify named entities in the complete test dataset. The training time is noted in each experiment as well as the number of support vectors and the accuracy measures achieved. Several sets of experiments are conducted, which include:

- Single-class experiments identifying protein names using the popular SVM implementation by Thorsten Joachims, SVM-Light [13-15], and different training data sizes.
- Multi-class experiments identifying all five named entities (protein, DNA, RNA, cell line, and cell type) using Joachims' All-Together multi-class implementation, SVM-Multiclass [9, 30], and different training data sizes.
- Single-class experiments identifying protein names only using the new SVM implementation, SVM-Perf [16, 17, 30, 31], and different training data sizes.
- The training data chunks range from 1,000 examples to 492,551 examples (the complete training dataset). Each set of experiments consists of 51 tests.
- Single-class experiments using SVM-Perf [16, 17, 30, 31] and varying values of the training error vs. margin error trade-off factor C. The training data size used for this round is fixed at 150,000 examples. The C-Factor values are reported in their SVM-Light equivalent for easier comparison and range from 0.01 to 1.0 in increments of 0.01. This set of experiments consists of 100 tests.
- All experiments use a linear kernel and a margin error of 0.1. The tests run on an Intel Core 2 quad-processor 2.66 GHz machine and a Xeon quad-processor 3.6 GHz machine. Running the same test on both machines completed in similar training time.

A. Single-Class Results

Using SVM-Light [13-15], a single-class support vector machine is trained to recognize protein name sequences. The trained machine is then used to classify proteins in the test data. Since no pre-processing was performed on the training and testing data besides features extraction, the positive examples in the data sets remained scarce. Training the SVM-Light machine with the complete training dataset and a trade-off value of 0.01 completed in about 28.5 minutes. The recall, precision, and F-score achieved in this case are 62.72, 56.12, and 59.23 respectively. Increasing the trade-off value to 1.0 raised the training time to about 269 minutes, and improved the accuracy measures to 68.92, 58.58, and 63.33 respectively. The SVM-Light curve in Fig. 2 summarizes the training time versus increased training data size. The training time is found to be polynomial $O(n^2)$ w.r.t. the training data size.

The same set of experiments is repeated using SVM-Perf [16, 17, 30, 31], which improves training time of linear machines to be linear w.r.t. the training data size. The training time improvement using SVM-Perf is several orders of magnitude as compared to that using SVM-Light, with the same classification results when trained with the same learning parameters. Fig. 2 compares the training time using both SVM-Light and SVM-Perf with the same data and learning parameters.

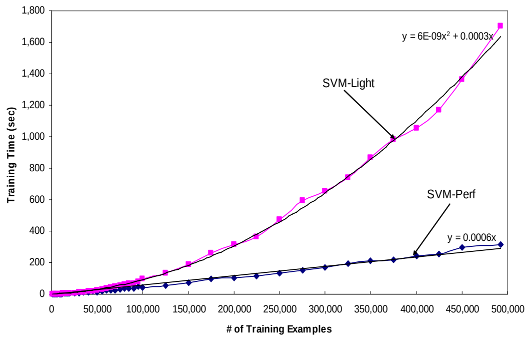

Fig. 2 Comparison of SVM-Light and SVM-Perf Training Time vs. Training Data Size

Training time is clearly enhanced to linear time using SVM-Perf. This is a major improvement in the field of support vector machines which makes training linear binary SVM and linear multi-class SVM approaches that are based on combining binary machines much more feasible and practical for real-world applications. This is particularly useful for text applications such as named entity recognition which have been shown to be mostly linear [33]. Using different learning parameters with both SVM-Light and SVM-Perf leads to the same conclusion, as the training time using SVM-Light remains polynomial $O(n^2)$ while being linear using SVM-Perf.

As a result of the observation of the enhanced accuracy measures with different training error vs. margin error trade-off factor values, and the associated impact on training time, a series of experiments investigating the effect of the trade-off factor C on training time is conducted using SVM-Perf. Since the training time is consistently longer with SVM-Light than it is with SVM-Perf, one can easily extrapolate the impact of the trade-off factor C on SVM-Light training time. The variation of training time w.r.t. the variation of the trade-off factor C is presented in Fig. 3. The training time scales in $O(\sqrt{C})$.

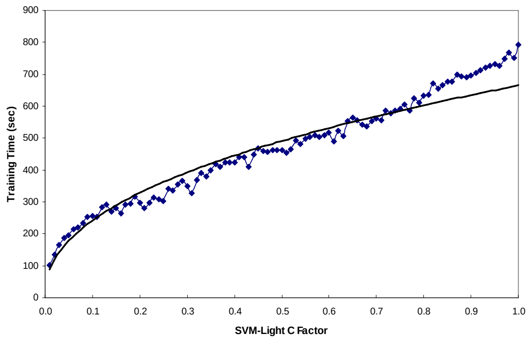

Fig. 3 SVM-Perf Training Time vs. Variation of C-Factor

As the variation of C clearly impacts the training time, its impact on various accuracy measures is investigated to assess whether it is possible to perform a grid search for tuning the learning parameters using a subset of the training dataset. A set of experiments using different training data sizes and a range of C values is conducted and the accuracy measures achieved are noted. TABLE II presents a sample of these results at training data size of 25,000, 250,000, and the complete set and some values of C. These experiments show that the accuracy improvement with varying C is consistent with different training data sizes, where the accuracy measures reach their best values within a very close range of C values. Using the same learning parameter for multi-class training reflects the same consistency in accuracy enhancement.

TABLE II
CONSISTENT PERFORMANCE CHANGE WITH VARYING C-FACTOR

Training Data Size	Trade-Off Factor C	Performance Recall/Precision/F-Score
25,000	C=0.01	32.72 / 38.55 / 35.40
	C=0.12	**53.74 / 48.87 / 51.19**
	C=0.14	53.51 / 48.66 / 50.97
	C=0.20	53.58 / 48.28 / 50.79
	C=1.00	52.29 / 47.54 / 49.80
250,000	C=0.01	58.32 / 53.68 / 55.90
	C=0.12	**69.11 / 60.09 / 64.29**
	C=0.14	69.05 / 60.01 / 64.21
	C=0.20	68.11 / 59.11 / 63.29
	C=1.00	65.03 / 56.88 / 60.69
492,551	C=0.01	62.72 / 56.12 / 59.23
	C=0.12	**73.04 / 62.48 / 67.34**
	C=0.14	72.72 / 62.16 / 67.03
	C=0.20	72.52 / 61.81 / 66.74
	C=1.00	68.06 / 59.29 / 63.37

This observation is very useful for tuning the learning parameters with a subset of the training dataset in order to achieve better accuracy with larger training data size during the final learning phase. Note that the F-score for protein single-class classification improved considerably using a *C* value of 0.12, with an eight points improvement over that achieved with *C* value of 0.01. Increasing the trade-off factor *C* any further led to a decline in the F-score measure, as observed in TABLE II. This shows that one can reach a reasonable trade-off between training time and the final accuracy measures by tuning the support vector machine with a fraction of the training dataset.

Comparing the accuracy measures improvement in the single-class training case (an F-score increase from 59.23 to 67.34), combined with the All-Together multi-class improved F-score of protein names from 59.23 to 65.90 using a *C* factor of 0.01 [11], leads one to believe that the potential accuracy improvement using All-Together multi-class training and a tuned value of the *C* factor would be a substantial motivation to use this technique with the proposed simplified machine learning technique for named entity recognition. The main prohibiting consideration in the time being is the extremely slow training time. An SVM-multiclass experiment with *C*=0.12 is currently being conducted to prove the potential enhancement of accuracy measures.

As part of the scalability experiments, the impact of the number of support vectors on training time is also observed as well as its relation to the training data size using both implementations. Fig. 4 and Fig. 5 depict the variation of the number of support vectors with the training data size using SVM-Light and SVM-Perf, respectively.

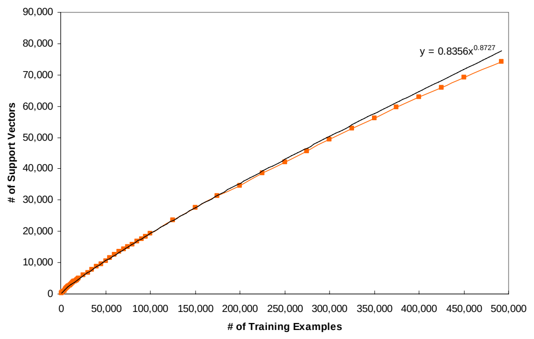

Fig. 4 SVM-Light Number of Support Vectors vs. Training Data Size

The number of support vectors using SVM-Light is $O(n^{0.8})$ w.r.t. the training data size. However, using SVM-Perf, the number of support vectors was only a very small fraction of the training data size and increased slightly with increased data size. The reduced number of support vectors is the main basis for the improved training time of SVM-Perf. The relationship between the number of support vectors and the training time is observed to be the same as that of the training data size, i.e., training time is polynomial w.r.t. the number of support vectors with SVM-Light, and linear with SVM-Perf.

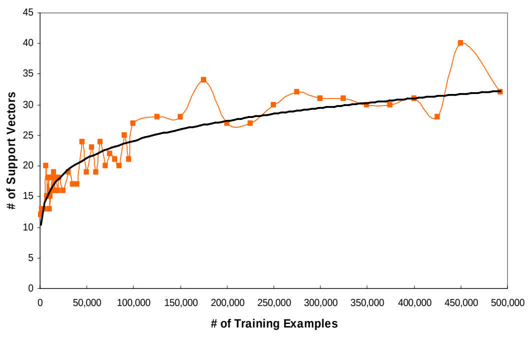

Fig. 5 SVM-Perf Number of Support Vectors vs. Training Data Size

B. Multi-Class Results

The SVM-Multiclass implementation by T. Joachims is based on [9] and uses a different quadratic optimization algorithm described in [30]. The SVM-Multiclass implementation uses an All-Together multi-classification approach, which is computationally more expensive yet usually more accurate than One-Against-All or One-Against-One multi-classification methods. Hsu and Lin [12] note that "as it is computationally more expensive to solve multi-class problems, comparisons of these methods using large-scale problems have not been seriously conducted. Especially for methods solving multi-class SVM in one step, a much larger optimization problem is required so up to now experiments are limited to small data sets." The multi-class experiments presented herein attempt to solve a real-world large-scale problem using an All-Together classification method. The training data is composed of 11 classes where each named entity is represented by two classes – one denoting the beginning of an entity and the other denoting a continuation token within the same entity – in addition to one class denoting non-named entity tokens.

Initial experiments for multi-class classification were unsuccessful, mostly due to hitting the processing power limits of the testing machines. The same experiments were attempted on different machine configurations, and unreasonably long processing time was needed to finally complete training using the complete training dataset. The first successful experiment required a total learning and classification time of 17 days in order to complete using a serial algorithm on a quad-processor Pentium IV machine. The same experiment was repeated on a Linux machine with four Xeon 3.6 GHz processors and completed in 97 hours or four days and one hour.

To explore the scalability issues of the All-Together multi-class SVM implementation, a series of experiments using different training data sizes is conducted with a low value for the *C* learning parameter equal to 0.01. The training time with 1,000 examples was 3.187 seconds and it increased considerably with increased data size to reach 416,264.251 seconds (6,937.738 minutes or 4.8 days) on the same machine.

The SVM-Multiclass [9, 30] implementation is based on the learning implementation in SVM-Light [13-15]. Fig. 6 reports the variation of the multi-class training time w.r.t. the training data size. The relationship remains polynomial $O(n^2)$ w.r.t. the training data size and a factor of $O(k^2)$ increase in time as compared to the single-class SVM-Light time, where k is the number of classes. Fig. 7 compares the training time of single-class training time using SVM-Light or SVM-Perf as compared to that of the multi-class implementation. It is clear that the training time required for All-Together multi-class training is prohibiting to using this approach with large datasets, unless new implementations are developed to address this issue.

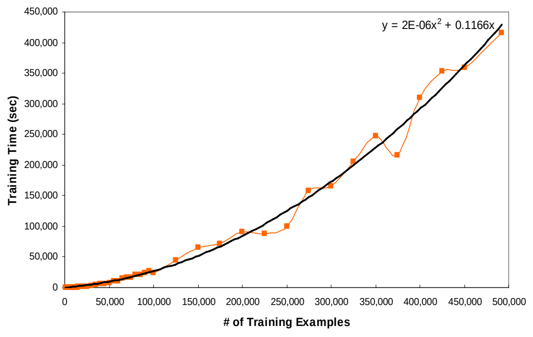

Fig. 6 SVM-Multiclass Training Time vs. Training Data Size

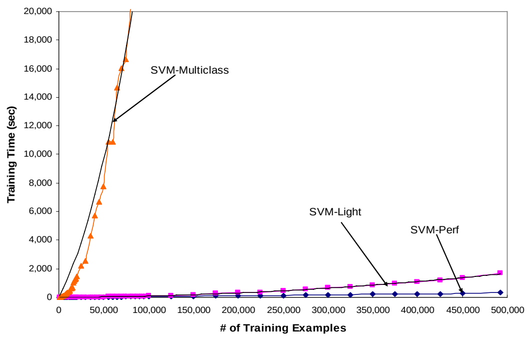

Fig. 7 Single and Multi-Class Training Time vs. Training Data Size

The observation of the number of support vectors (SV) produced by SVM-Multiclass in relationship to the training data size (Fig. 8) shows that the number of SVs is $O(n^{0.8})$ w.r.t. the training data size. This is the same relationship observed with the number of SVs produced in the single-class case using SVM-Light. Since SVM-Multiclass uses the same learning modules of SVM-Light, one may anticipate a potential scalability improvement by developing a new All-Together multi-class implementation based on the improved learning time of SVM-Perf.

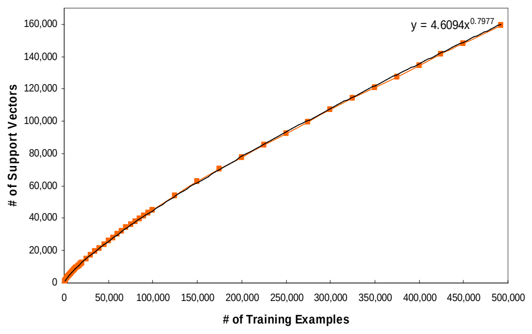

Fig. 8 SVM-Multiclass Number of Support Vectors vs. Training Data Size

Fig. 9 presents the impact of the training data size on the multi-class classification performance measures in terms of precision, recall, and $F_{\beta=1}$-score. It is noted that the performance measures improve sharply with increased training data size in the low range of sizes, while only slight improvement, if any, is achieved with further increase in training data size. The trade-off between the potential performance enhancement vs. the cost of training in terms of computational time and memory needs may need to be considered when applying the All-Together multi-classification technique to large datasets.

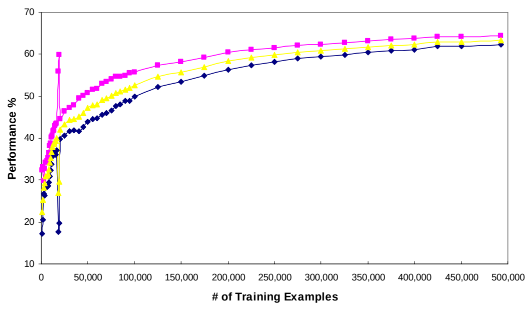

Fig. 9 SVM-Multiclass Overall Performance vs. Training Data Size From Top to Bottom: Precision, F-Score, and Recall

Finally, a comparison of the training time of the All-Together experimental results to estimates of the training time using approaches based on combining binary SVM classifiers – namely One-Against-All, One-Against-One, and Half-Against-Half approaches – is presented. Fig. 10 compares the experimental multi-class experimental results to estimates of the training time using the other approaches based on SVM-Light single-class results with the same learning parameters. The comparison weighs against using the All-Together approach despite the potential accuracy enhancement, until a new implementation is available to improve its scalability.

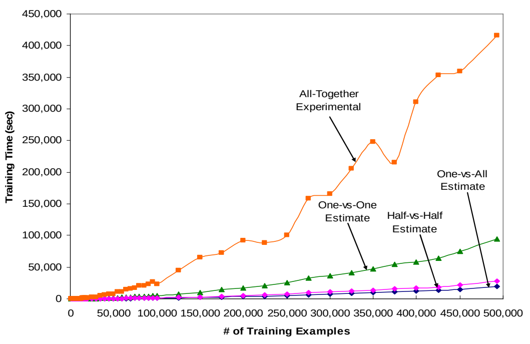

Fig. 10 Multi-Class Training Time Using Various Approaches Estimates Based on SVM-Light Binary Training Time

In order to investigate the potential improvement in training for multi-class learning using the learning algorithms implemented in SVM-Perf [17] for training linear machines, an initial prototype is developed and the preliminary experiments using this prototype resulted in a tremendous improvement of the training time while achieving the same accuracy measures as SVM-Multiclass. The new prototype implementation is referred to as SVM-PerfMulti. A sample of the training time results is reported in TABLE III. These initial scalability improvement results are very promising and a complete analysis is currently undergoing in order to implement the full multi-class learning solution for linear machines. The initial prototype improves the training time at the expense of increased memory needs. Further investigation is needed in order to improve the memory consumption.

TABLE III
COMPARISON OF SVM-PERFMULTI AND SVM-MULTICLASS TRAINING TIME VS. TRAINING DATA SIZE

Training Data Size	**SVM-Multiclass Training Time (seconds)**	**SVM-PerfMulti Training Time (seconds)**
5,000	94.213	3.028
10,000	355.101	10.232
20,000	1,453.887	39.689
50,000	7,784.148	175.413
100,000	23,531.920	518.839
200,000	91,632.975	1,624.452
300,000	165,178.301	3,522.994

V. CONCLUSION AND FUTURE WORK

In this paper, a series of experiments is presented in order to explore the scalability issues associated with solving the named entity recognition problem using multi-class support vector machines and high-dimensional features. Baseline experiment results have shown that the proposed language and domain-independent approach is capable of successfully recognizing and classifying named entities with reasonable accuracy measures. These measures are further improved by tuning the learning parameters at the expense of longer training time. The potential improvement in the classification accuracy measures using an All-Together multi-class training and tuned parameters constitute a motivation to investigate ways to improve the scalability of this technique.

The new implementation of binary SVM classifiers, SVM-Perf, offers a great improvement in training time with consistent accuracy performance measures compared to those obtained using the currently popular SVM-Light implementation. We strongly recommend switching to SVM-Perf for training linear SVM machines. An initial prototype focused on improving scalability of the All-Together multi-class technique using the learning algorithms of SVM-Perf results in a tremendous improvement in the training time but increases the memory needs. The preliminary results are encouraging and further analysis and implementation of the improved All-Together multi-class learning and classification is currently undergoing.

REFERENCES

[1] S. Abe, Support Vector Machines for Pattern Classification. London: Springer-Verlag, 2005.
[2] E. Alpaydin, Introduction to Machine Learning. Cambridge, MA: The MIT Press, 2004.
[3] T. Ban and S. Abe, "Spatially Chunking Support Vector Clustering Algorithm," in Proc. of the IEEE International Joint Conference on Neural Networks, Grenoble, France, 2004.
[4] M. Barros de Almeida, A. de Padua Braga, et al., "SVM-KM: Speeding SVMs Learning with A Priori Cluster Selection and K-Means," in Proc. of the 6th Brazilian Symposium on Neural Networks, 2000.
[5] K. P. Bennett and C. Campbell, "Support Vector Machines: Hype or Hallelujah?," SIGKDD Explor. Newsl., vol. 2, pp. 1-13, 2000.
[6] D. Boley and D. Cao, "Training Support Vector Machine using Adaptive Clustering," in Proc. of the 4th SIAM International Conference on Data Mining, Lake Buena Vista, Florida, 2004.
[7] R. Collobert, F. Sinz, et al., "Large Scale Transductive SVMs," Journal of Machine Learning Research, pp. 1687-1712, 2006.
[8] R. Collobert, F. Sinz, et al., "Trading Convexity for Scalability," in Proc. of the 23rd international conference on Machine learning, Pittsburgh, PA, 2006.
[9] K. Crammer and Y. Singer, "On the Algorithmic Implementation of Multi-class SVMs," Journal of Machine Learning Research, vol. 2, pp. 265–292, 2001.
[10] C. Giuliano, A. Lavelli, et al., "Simple Information Extraction (SIE)," ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica, 2005.
[11] M. S. Habib and J. Kalita, "Language and Domain-Independent Named Entity Recognition: Experiment using SVM and High-Dimensional Features," in Proc. of the 4th Biotechnology and Bioinformatics Symposium (BIOT-2007), Colorado Springs, CO, 2007.
[12] C.-W. Hsu and C.-C. Lin, "A Comparison of Methods for Multi-Class Support Vector Machines," IEEE Transactions on Neural Networks, vol. 13, pp. 415-425, 2002.
[13] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in Proc. of the European Conference on Machine Learning, 1998.
[14] T. Joachims, "Making Large-Scale SVM Learning Practical," in Advances in Kernel Methods - Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola, Eds.: MIT-Press, 1999.
[15] T. Joachims, Learning to Classify Text Using Support Vector Machine. Norwell, MA: Kluwer Academic, 2002.
[16] T. Joachims, "A Support Vector Method for Multivariate Performance Measures," in Proc. of the International Conference on Machine Learning (ICML), 2005.

International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:1, 2008

[17] T. Joachims, "Training Linear SVMs in Linear Time," in Proc. of the ACM Conference on Knowledge Discovery and Data Mining (KDD), 2006.
[18] V. Kecman, Learning and Soft Computing. London, UK: The MIT Press, 2001.
[19] J. D. Kim, T. Ohta, et al., "GENIA Corpus--Semantically Annotated Corpus for Bio-Textmining," Bioinformatics, vol. 19 Suppl 1, pp. 180-182, 2003.
[20] J.-D. Kim, T. Ohta, et al., "Introduction to the Bio-Entity Recognition Task at JNLPBA," in Proc. of the 2004 Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA'2004), Geneva, Switzerland, 2004.
[21] U. H.-G. Kreßel, "Pairwise Classification and Support Vector Machines," in Advances in Kernel Methods: Support Vector Learning. Cambridge, MA: MIT Press, 1999, pp. 255-268.
[22] K.-J. Lee, Y.-S. Hwang, et al., "Biomedical Named Entity Recognition using Two-Phase Model Based on SVMs," Journal of Biomedical Informatics, vol. 37, pp. 436-447, 2004.
[23] H. Lei and V. Govindaraju, "Half-Against-Half Multi-class Support Vector Machines," in Proc. of the 6th International Workshop on Multiple Classifier Systems, Seaside, CA, USA, 2005.
[24] K.-R. Müller, S. Mika, et al., "An Introduction to Kernel-Based Learning Algorithms," IEEE Transactions on Neural Networks, vol. 12, pp. 181-120, 2001.
[25] K.-M. Park, S.-H. Kim, et al., "Incorporating Lexical Knowledge into Biomedical NE Recognition," in Proc. of the 2004 Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA'2004), Geneva, Switzerland, 2004.
[26] J. C. Platt, N. Cristianini, et al., "Large Margin DAGs for Multiclass Classification," in Advances in Neural Information Processing Systems, vol. 12, S. A. Solla, T. K. Leen, and K.-R. M¨uller, Eds. Cambridge, MA: MIT Press, 2000, pp. 547-553.
[27] M. Rössler, "Adapting an NER-System for German to the Biomedical Domain," in Proc. of the 2004 Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA'2004), Geneva, Switzerland, 2004.
[28] T. Serafini and L. Zanni, "On the Working Set Selection in Gradient Projection-based Decomposition Techniques for Support Vector Machines," Optimization Methods and Software, pp. 583-596, 2005.
[29] Y. Song, E. Kim, et al., "POSBIOTM-NER in the Shared Task of BioNLP/NLPBA 2004," in Proc. of the 2004 Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA'2004), Geneva, Switzerland, 2004.
[30] I. Tsochantaridis, T. Hofmann, et al., "Support Vector Learning for Interdependent and Structured Output Spaces," in Proc. of the 21st International Conference on Machine Learning (ICML), Alberta, Canada, 2004.
[31] I. Tsochantaridis, T. Joachims, et al., "Large Margin Methods for Structured and Interdependent Output Variables," Journal of Machine Learning Research (JMLR), vol. 6, pp. 1453-1484, 2005.
[32] V. N. Vapnik, Statistical Learning Theory. New York, NY: John Wiley & Sons, 1998.
[33] Y. Wong and H. T. Ng, "One Class per Named Entity: Exploiting Unlabeled Text for Named Entity Recognition," in Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), Hyderabad, India, 2007.
[34] G. Zhou and J. Su, "Exploring Deep Knowledge Resources in Biomedical Name Recognition," in Proc. of the 2004 Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA'2004), Geneva, Switzerland, 2004.