Adapting SVM for Natural Language Learning: A Case Study Involving Information Extraction

Yaoyong Li, Kalina Bontcheva, Hamish Cunningham
Department of Computer Science, The University of Sheffield, UK

Abstract

Support Vector Machines (SVM) have been used successfully in many Natural Language Processing (NLP) tasks. This paper investigates two techniques to help SVM deal with the two unique features of NLP problems, namely imbalanced training data and the difficulty of obtaining sufficient training data. Firstly, the uneven margins SVM is introduced as an adaptation of the standard SVM model for imbalanced training data. Secondly, SVM active learning is investigated, since it achieves a given performance level with fewer training examples than conventional passive learning. The two algorithms were evaluated on several Information Extraction (IE) tasks, where both achieved better performance than the standard SVM and the SVM with passive learning, respectively. Moreover, by combining the uneven margins SVM with the active learning algorithm, we achieved the best results on the Seminars corpus, a benchmark dataset for IE. The uneven margins SVM also obtained the best results on another IE benchmarking dataset – the Jobs corpus. Due to the similarities between IE and other NLP tasks, the two techniques are likely to be beneficial in a wide number of applications. In addition, we also compare several approaches of converting multi-class problem into binary classification problems with respect to the standard SVM and the uneven margins SVM, respectively. We found that one approach achieved best results on the three datasets when using the uneven margins SVM, which is also computationally more efficient than others.

1 Introduction

Support Vector Machines (SVM) is a supervised machine learning algorithm, which has achieved state-of-the-art performance on many learning tasks. In particular, SVM is a popular learning algorithm for Natural Language Processing (NLP) tasks such as POS (Part-of-speech) tagging [22, 31], word sense disambiguation [25], NP (noun phrase) chunking [23], information extraction[21, 27], relation extraction [45], semantic role labeling [19], and dependency analysis [24, 43]. Almost all these applications adopt the same steps: first they transform the problem into a multi-class classification task; then convert the multi-class problem into several binary classification problems; then an SVM classifier is trained for each binary classification[1]; and finally, the classifiers' results are combined to obtain

[1]The SVM has been formulated as multi-class classifier in various forms so that it can solve multi-class problem directly (see e.g. [8]), Recently researchers (see e.g. [39]) have applied the multi-class SVM to some NLP problems and obtained better results than using binary SVM. Therefore, applying multi-class SVM to multi-class problem is a promising research area. On the other hand, the implementation of the multi-class SVM is more complicated than that of binary SVM. [20] compared the binary SVM with one form of multi-class SVM presented in [8] on several standard machine learning datasets and their results showed no clearly difference of the performances between binary SVM performed and multi-class SVM. More investigations are needed for evaluating multi-class SVM for NLP applications. This paper only considers the binary SVM classifier, which has been used in most applications.

the solution to the original NLP problem.

SVM is an optimal classifier in the sense that, given training data, it learns a classification hyperplane in the feature space which has the maximal distance (or margin) to all the training examples (except a small number of examples as outliers) (see e.g. [9]). Consequently, on classification tasks the SVM tends to have better generalisation capability on unseen data than other distance- or similarity-based learning algorithms such as k-nearest neighbour (KNN) or decision trees. Another feature of the SVM is that, by using different types of kernel function, the SVM can explore different kinds of combinations of the given features without increasing computational complexity. In contrast, it would be difficult for many other learning algorithms to deal with a huge number of feature combinations efficiently.

On the other hand, NLP tasks typically represent instances by very high dimensional but very sparse feature vectors, which leads to positive and negative examples being distributed into two distinctly different areas of the feature space. This is particularly helpful for the SVM to search a classification hyperplane in feature space and for the generalisation capability of the classifier as well. That is a main reason why the SVM can achieve very good results in a variety of NLP tasks. Such very high dimensional representation is achieved by forming the feature vector explicitly from text using a huge amount of linguistic features and in many cases by exploring the so-called kernel function to map the feature vector into higher dimensional space (see below).

Furthermore, as the SVM is optimal margin classifier, the distance of an example to the SVM classification hyperplane indicates how important of the example to the SVM learning. The examples being close to the SVM hyperplane are crucial for the learning. The SVM active learning is based on the distance of unlabelled example to the SVM hyperplane (see e.g. [2])

In the applications, the SVM can select the useful features effectively from a large number of features for a particular classification problem. This is because the SVM learns a classifier by combining all the features (or feature combinations) with different weights. If a feature occurs almost equally in both positive and negative training examples and hence is irrelevant to the classification, its learned weight would have little contribution to the classifier. In contrast, many other algorithms require careful manual feature selection. This is advantageous when applying the SVM to NLP problem. As there exists many types of NLP features from morphology, syntax, semantics as well as from different knowledge sources like thesaurus and gazetteers, in the application of SVM to NLP, those different kinds of features are just put together to form one feature vector for one example as input to SVM and the learning would automatically determine which features and/or combinations of features are useful for the task.

When compared to other classification problems, NLP classification tasks have several unique characteristics, which were seldom considered in applications. Perhaps the most important one is that NLP tasks tend to have imbalanced training data, in which positive examples are vastly outnumbered by negative ones. This is particularly true for smaller data sets where often there are thousands of negative training examples and only few positive ones. Another unique characteristic is that annotating text for training the algorithm is a time-consuming process, while at the same time unlabelled data is abundant.

Moreover, since the SVM is usually used as binary classifier for solving an NLP problem and an NLP problem in most case is equivalent to a multi-class classification problem, we need to transform the multi-class problem into binary classification problems. Several methods have been proposed for the transformation. [20] compared some of those methods using standard SVM model on some standard problems for machine learning. But none of

their problems was for NLP.

Therefore, when the SVM is applied to NLP tasks, these particular aspects should be taken into account in order to obtain a practical system with good performance. In this paper we study two techniques to adapt SVM to NLP problems: *(i)* use an uneven margins SVM model (see [29]) to deal with imbalanced training data; *(ii)* employ SVM active learning to alleviate the difficulty in obtaining labeled training data. The algorithms are presented and tested on several Information Extraction (IE) tasks, however we believe that they could also improve SVM performance on other NLP tasks. We also evaluate different methods of converting multi-class problem into binary classification problems for the NLP applications and particularly for the uneven margins SVM.

The rest of paper is structured as follows. Section 2 introduces the IE tasks and describes in further detail the unique characteristics of NLP applications in the context of IE. We present our classifier based IE framework, the uneven margins SVM and SVM active learning for IE in Section 3. Section 4 tests the algorithms and the framework on three benchmarking corpora for IE. Section 5 discusses related work and Section 6 summarises our findings.

2 Machine Learning for Information Extraction

Information Extraction (IE) is a technology based on analysing natural language in order to extract snippets of information. The process takes texts (and sometimes speech) as input and produces fixed-format, unambiguous data as output, e.g., events, entities or relations can be extracted automatically from text such as newswire articles or Web pages. IE is useful in many applications, such as information gathering in a variety of domains, automatic annotations of web pages for Semantic Web, and knowledge management.

A wide range of machine learning techniques have been used for IE and achieved state-of-the-art results, comparable to manually engineered IE systems. The learning algorithms for IE can be classified broadly into two main categories: rule learning and statistical learning ones. The former induce a set of rules from training examples. There are many such rule-based learning systems, e.g. SRV [15], RAPIER [1], WHISK [36], BWI [17], and $(LP)^2$ [5]. Statistical systems learn a statistical model or classifiers, such as HMMs [14], Maximal Entropy [4], SVM [21, 30], and Perceptron [3].

IE systems also differ from each other in the NLP features that they use. These include simple features such as token form and capitalisation information, linguistic features such as part-of-speech, semantic information from gazetteer lists, and genre-specific information such as document structure. In general, the more features the system uses, the better performance it can achieve.

Since we uses binary SVM classifier, we concentrate on classifier-based learning for IE, which typically converts the recognition of each information entity into a set of classification problems. There exists several methods for using binary classifier for solving multi-classes problem. We will describe in detail those methods in Section 3.1 and experimentally compare them in Section 4.2. We are particularly interested in one method which was rarely used in the NLP applications of SVM in comparison with other methods. It trains two binary classifier for each type of information entity: one for recognising the entity's start token and the other for the entity's end token. It need less computing resources than other methods. Our experiments showed that it also has better performances than other methods when using the uneven margins SVM model.

The classification problem derived from IE usually has imbalanced training data, which is

particularly true for smaller data sets where often there are thousands of negative training examples and only few positive ones. Two approaches have been studied so far to deal with imbalanced data in IE. The first one under-samples the majority class or over-samples the minority class in order to obtain a relatively balanced training data [44]. However, under-sampling can potentially remove certain important examples, and over-sampling can lead to over-fitting and a larger training set. The second approach is to divide the problem into several sub-problems in two layers, each of which has less imbalanced training set than the original one [3, 35]. The output of the classifier in the first layer is used as the input to the classifiers in the second layer. As a result, this approach needs more classifiers than the original problem. Moreover, the classification errors in the first layer will affect the performance of the second one.

In this paper we explore another approach to handling the imbalanced data in IE, namely, adapting the learning algorithm for balanced classification to deal better with imbalanced data. In particular, we use the uneven margins SVM model for imbalanced training sets.

In addition to the problem with imbalanced training sets, there is also the problem of obtaining sufficient training data for IE. In general, a learning algorithm derives a model from a set of documents which have been manually annotated by the user. Then the model can be used to extract information from new documents. However, manual annotation for IE is a labour-intensive and time-consuming process due to the complexity of the task. Hence, frequently the machine learning system trains only on a small number examples, which are selected from a pool of unlabelled documents.

One way to overcome this problem is to use *active learning* which minimises the number of labeled examples required to achieve a given level of performance. It is usually implemented as a module in the learning system, which selects an unlabelled example based on the current model and/or properties of the example. Then the system asks the user to label the selected example, adds the new labeled example into the training set, and updates the model using the extended training set. Active learning is particularly useful in IE, where there is an abundance of unlabeled text, among which only the most informative instances need to be found and annotated.

SVM active learning is an SVM-specific algorithm [2, 34], which uses the margin (or the distance) from the unlabelled example to the classification hyperplane as a measure for the importance of the example for learning. SVM active learning has been applied successfully in applications, such as text classification [38], spoken language understanding [40], named entity recognition [41] and Japanese word segmentation [33]. These experiments have shown that SVM active learning clearly outperformed the SVM with passive learning. In this paper we investigate SVM active learning for IE tasks, with focus on the algorithm settings which are specific to NLP tasks. We also explore the uneven margins SVM model for active learning. In contrast, previous works on the SVM active learning only used the standard SVM model.

3 SVM Learning for IE

3.1 The SVM Based IE System

Classifier-based framework. The classifier-based framework we adopted for applying SVM to IE consists of three stages: pre-processing of the documents to obtain feature vectors, learning classifiers or applying classifiers to test documents, and finally post-processing the results to tag the documents.

Since we consider the IE as a multi-class classification problem where each token in text is check to see whether or not it belong to an entity with a particular type, and we use the SVM as a binary classifier, we need to adopt some strategy to transform the multi-class problem to binary classification problems. The key part of the framework we use is to convert the recognition of each type of information entity into two binary classification tasks — one to decide whether a token is the start of an entity and another one for the end token. In contrast, previous work adopted slightly different frameworks for SVM classification for IE. For instance, [21] trained four SVM classifiers for each entity type – besides the two SVMs for start and end (like ours), also one for middle tokens, and one for single token entities. They also trained an extra SVM classifier to recognise tokens which do not belong to any named entity. In other words, they learned one classifier for each part of entity and for non-entity. Another approach is to train an SVM classifier for every possible transition of tags [30]. Depending on the number of entities, this approach may result in a large number of SVM classifiers. Hence, in comparison, our approach is simpler than the other two SVM-based systems, in terms of requiring the lowest number of SVM classifiers.

In fact there are two main frameworks for applying the SVM or other binary classifier to the IE and other NLP tasks. They can be seen as learning one classifier for each type of tag (or pair of tags) in two different schemes of tagging entities in text, respectively. One tagging scheme assigns every token a tag. The possible tag of one token is the beginning, middle or end token of a multi-token entity with particular type, or the single token of a single-token entity, or non-entity (or other)token which does not belong to any entity. We call this tagging scheme as *BMESO* scheme. Alternatively, we can only tag the beginning and end tokens of entity, and regard the token in single-token entity as both the beginning and end token of the entity. We call this tagging scheme as *BE* scheme. Correspondingly, one framework of converting multi-class problem into binary classification problems learns a classifier for each type of tag (or pair of tags) in the BMESO tagging scheme. Another framework only learns the classifiers for the beginning or end token of entity of certain type in the BE tagging scheme. For the sake of convenience, we call the first framework as *BMESO framework* and the second as *BE framework*.

In the implementation of the BMESO framework, two approaches have been used. Suppose there are K classes in the multi-class problem, one approach, called *one vs. others*, learns K classifiers each of which separates one class from all others. Another approach is called *pairwise classification* that learns $K(K-1)/2$ binary classifiers corresponding to all pairs of classes. In the application of the learned classifiers, the one vs. others approach compares the outputs of all classifiers (before thresholding) for one instance and assign the instance the class which classifier has the maximal output. In contrast, in the pairwise approach, for one instance, one classifier votes for one class or another class according to its output and final class decision for the instance is made by majority voting.

Most of previous works in the application of SVM to NLP used the one vs. others approach in the BMESO framework. For instance, [22], [21] and [45] adopted the framework for POS tagging, IE and relation extraction, respectively. A few works used the pairwise method, such as [23] for NP chunking. [27, 18] adopted the BE framework in the application of the SVM to IE. [17, 5] also used the BE framework for IE but they both explored rule learning algorithms rather than statistical learning such as the SVM. [20] compared the one vs. others and the pairwise method with some other method, but not including the BE framework. It used the standard SVM model on some standard problems for machine learning. But none of their problem was from NLP.

One advantage of the BE framework over the BMESO framework is that the BE framework learns less classifier, resulting in less computation time and less computer memory for storing learned model. Suppose the SVM is applied to an IE task with N types of entity. In the BMESO framework it is a multi-class problem with $4 * N + 1$ classes. It needs learn

$4 * N + 1$ SVM classifiers if using the one vs. other method, and learn $(4 * N + 1) * (4 * N)/2$ classifiers if using pairwise method. In contrast, the BE framework only need learn $2 * N$ SVM classifiers. In other words, the one vs. other method needs more than twice classifiers of the BE method. The pairwise method needs much more classifiers than other two methods if n is not very small.

Another advantage is that the BE framework may achieve better results, because the framework selected some specific labels for learning, which are more characteristic than other labels so that the resulting classification problems may be easier than the problems for other labels. For example, for multi-token entity, the start and end tokens may not only make contribution to the meaning of the entity, but also have the responsibility to distinguish the entity from the surrounding tokens[2]. In contrast, a middle token just bear the meaning of the entity. Therefore, the boundary tokens of an entity is more characteristic for the type of the entity than the middle token(s). Consequently the classifier learned for the boundary token have better generalisation capability than the classifier for middle token[3].

In Section 4.2 we will experimentally compare the two frameworks by using the standard SVM and the uneven margins SVM, respectively.

Pre- and post-processing. The aim of the pre-processing is to form input vectors from documents. In our IE system each document is first processed using the open-source ANNIE system, which is part of GATE[4] [10]. This produces a number of linguistic (NLP) features, including token form, capitalisation information, token kind, lemma, part-of-speech (POS) tag, semantic classes from gazetteers, and named entity types according to ANNIE's rule-based recogniser.

Based on the linguistic information, a feature vector is constructed for each token, as we iterate through the tokens in each document (including word, number, punctuation and other symbols) to see if the current token belongs to an information entity or not. The feature vector derived from one token is binary vector, each dimension of which corresponds to one of those features collected from training set and each binary component indicates whether or not the corresponding feature exists for the token. Hence, the feature vector is very high dimensional but very sparse because the total number of features is very large and the number of features one particular token has is quite small. For example, one feature vector for the Seminars corpus used in our experiments had approximately 20000 dimensions but only at most 7 non-zero components.

Since in IE the context of the token is usually as important as the token itself, the SVM input vector needs to take into account features of the preceding and following tokens, in addition to those of the given token. In our experiments the same number of left and right tokens was taken as a context. In other words, the current token was at the centre of a window of tokens from which the features are extracted. This is called a window size. Therefore, for example, when the window size is 3, the algorithm uses features derived from 7 tokens: the 3 preceding, the current, and the 3 following tokens. Due to the use of a context window, the SVM input vector is the combination of the feature vector of the current token and those of its neighboring tokens. We concatenated the feature vector of the current token and the feature vectors of the surrounding tokens as the input vector to the SVM.

[2] In comparison with English or other western language, the function of token for separating the entity from surrounding token is more important in oriental languages such as Chinese, Japanese or Korean where there is no delimiter between two consecutive words in a sentence. In fact, the major function of some tokens in these languages is to separate one entity from the preceding or following entity.

[3] In the case of Chinese, [28] showed that the classifiers for the start and end tokens of Chinese word has clearly better performances than the classifier for middle token.

[4] Freely available from http://www.gate.ac.uk/

As the input vector incorporates information from the context surrounding the current token, features from different tokens can be weighted differently, based on their position in the context. The weighting scheme we used was the *reciprocal scheme*, which weights the surrounding tokens reciprocally to the distance to the token in the centre of the context window. This reflects the intuition that the nearer a neighbouring token is, the more important it is for classifying the given token. Our experiments showed that such a weighting scheme obtained better results than the commonly used equal weighting [27].

After classification, the start and end tags of the entities are obtained and need to be combined into one entity tag. Therefore some post-processing is needed to guarantee tag consistency and to try to improve the results by exploring other information. The currently implemented procedure has three stages. First, in order to guarantee the consistency of the recognition results, a document is scanned from start to end to remove start tags without matching end tags and end tags without preceding start tags. The second stage filters out candidate entities from the output of the first stage, based on their length. Namely, a candidate entity tag is removed if the entity's length (i.e., the number of tokens) is not equal to the length of any entity of the same type in the training set. The third stage puts together all possible tags for a sequence of tokens and chooses the best one according to the probability which was computed from the output of the classifiers (before thresholding) via a Sigmoid function. See [27] for details, which also showed that the system's performance was improved significantly after the first stage of post-processing and improved slightly after the second and third stages.

3.2 Uneven Margins SVM

A binary SVM classifier corresponds to a hyperplane in feature space with maximal margin, which would separate positive training examples from negative ones. The margin is the distance from the training examples to the hyperplane. The margin can be regarded as a measure of the error-tolerance ability of the classifier, since a classifier is more likely to classify a test instance correctly if it has a larger margin. Generally, if a training set is representative of the whole dataset, a classifier with a larger margin with respect to the training set would have a better generalisation performance. However, if the training set is unrepresentative, we should take great care of the margin in the margin learning algorithms such as SVM, because the maximal margin classifier learned from an unrepresentative training set may have poor generalisation performance, as illustrated in Figure 1.

Figure 1 shows a toy 2-dimensional binary classification problem together with two kinds of training sets and the corresponding SVM classifiers. The training examples in the left part of Figure 1 are representative of the whole dataset, and therefore the maximal margin classifier learned from the training set can classify correctly most of the unseen test data, meaning that the SVM classifier has a good generalisation capability. In contrast, the right graph illustrates a situation where the training set is not representative of the distribution of all positive examples due to the very small number of available training examples (only three). In this case, the SVM classifier with maximal margin would mistakenly classify many unseen positive examples as negative ones. Unfortunately, many imbalanced classification problems, such as those arising in IE, have quite small number of positive training examples, resulting in an SVM classifier with poor generalisation capability. As a matter of fact, previous work has demonstrated that SVM classifiers trained on imbalanced training data have poor generalisation performance (see e.g. [26, 29])

However, as can be seen in Figure 1, if the classification hyperplane could be moved away from the positive training examples in imbalanced dataset, then the classifier would classify more unseen data correctly, i.e., it would have better generalisation performance. There-

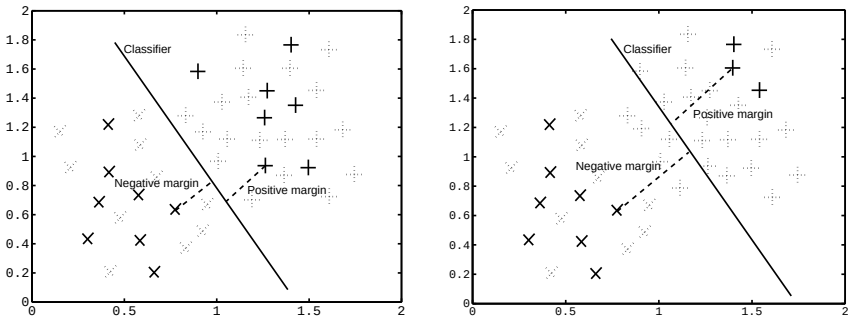

Figure 1: A toy 2-dimensional classification problem and two SVM classifiers. The two graphs illustrate two different kinds of training sets. The training set on the left is representative of the whole dataset, whereas the positive examples in the training set on the right are not. In both figures a '+' represents a positive example and a 'x' – a negative example. The solid line '+' and 'x' are the training examples and those with dashed lines are the test ones.

fore, if an SVM classifier has to be learned from an imbalanced training set which has only a few positive examples, it may be beneficial to require the learning algorithm to set the margin with respect to the positive examples (the positive margin) to be somewhat larger than the margin with respect to the negative examples (the negative margin). In other words, in order to achieve better generalisation performance, one needs to distinguish the positive margin from the negative margin when training the SVM. Therefore, we introduced a margin parameter τ into the SVM optimisation problem to control the ratio of the positive margin over the negative margin (for details see [29]).

Formally, given a training set $\mathbf{Z} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$,where $\mathbf{x}_i$ is the n-dimensional input vector and y_i ($= +1$ or -1) its label, the SVM with uneven margins is obtained by solving the quadratic optimisation problem:

$$\min_{\mathbf{w},\, b,\, \xi} \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{m} \xi_i$$

$$\begin{aligned} \text{s.t.} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i + b \geq 1 \quad & \text{if } \; y_i = +1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle - \xi_i + b \leq -\tau \quad & \text{if } \; y_i = -1 \\ \xi_i \geq 0 \qquad & \text{for } \; i = 1, ..., m \end{aligned}$$

where a parameter τ was added to the constraints of the optimisation problem for the standard SVM formation. τ is the ratio of negative margin to the positive margin of the classifier. It is equal to 1 in the standard SVM, which treats positive and negative examples equally. However, for imbalanced training data (as is the case with IE), where the positive examples are so rare that they are not representative of the genuine distribution of all positive examples, a larger positive margin than negative one (namely $\tau < 1$) would be beneficial for the generalization of the SVM classifier.

When applying the uneven margins SVM to a problem, we first have to determine a value for the uneven margins parameter τ. If the problem has just a few positive training examples and many negative ones, then $\tau < 1$ would be helpful. However, the optimal value of τ is not entirely dependent upon the number of positive examples in the training set — instead it is actually dependent upon the distribution of positive training examples among all positive examples. Like other parameters of learning algorithms, the value of τ can be empirically

determined by, for example, n-fold cross-validation on the training set or using a hold-out development set. On the other hand, Table 7 in Section 4.3 shows that the performance of the uneven margins SVM is robust with respect to the value of the uneven margins parameter, probably because the SVM classifier was learned in a very high dimensional space where the positive training examples may possibly be far away from the negative ones. Therefore, a reasonable estimation of τ is able to help the uneven margins SVM to achieve significantly better results than the standard SVM model.

As showed in [29], for the SVM model with bias term, the solution of uneven margins SVM can be obtained from a related standard SVM via a transformation. The transformation is simple — basically it amounts to adding a τ related term to the bias term b of the corresponding standard SVM model. Therefore, in order to achieve computational gains, the uneven margins SVM problem is not solved directly. Instead, a corresponding standard SVM problem is solved first by using an existing SVM implementation, (e.g., a publicly available SVM package[5]), and then the solution of uneven margins SVM is obtained through a transformation.

On the other hand, the transformation means that the uneven margins SVM is the same as the standard SVM except for a shift of bias term or equivalently, a shift of SVM's output before thresholding. Note that in our experiments we use the same value of uneven margin parameter for all the SVM classifiers computed. Hence all the classifiers' outputs have the same amount of shift with respect to the uneven margin parameter. As a result, the uneven margins SVM would have the same behaviour as the standard SVM if used in the one vs. others approach of the BMESO framework (see Section 3.1). This is because the class decision in the one vs. others approach is made by comparing the outputs of all classifiers (before thresholding) which is not effected by the same amount of shift on the outputs of all the classifiers resulted from using the uneven margins parameter. In contrast, the uneven margins SVM does have effect when used in the *BE framework*, or *pairwise classification* approach, because they both take into account binary output of every individual SVM classifier and the shift of the SVM's output before thresholding does make difference in the binary output.

3.3 SVM Active Learning for IE

Apart from handling imbalanced training data, the margin of the SVM classifier to one individual example can also be used for measuring how informative the example is for training the SVM classifier – this forms the basis for the *SVM active learning* algorithm.

Given an SVM classifier (in primal form) $W = \{w_1, \ldots, w_l\}$ and b and an example $X = \{x_1, \ldots, x_l\}$, the margin of the example X to the SVM is as

$$m(X, W) =< X, W > +b = \sum_{i=1}^{l} x_i w_i + b \quad (1)$$

which measures how close the example is to the hyperplane and can be regarded as a confidence of the SVM classifying X. The smaller the margin $m(X, W)$ is, the less confidence the SVM has for classifying the example X. In other words, the smaller the margin of an example to the SVM classifier is, the more informative the example might be for training the SVM. Therefore, the SVM active learning algorithm is based on the margin – it selects the example(s) with the smallest margin (least confidence).

[5]The SVM^{light} package, available from http://svmlight.joachims.org/, was used to learn the SVM classifiers in our experiments.

SVM active learning has been successfully used for text categorisation and image retrieval, which are essentially classification problems. Since IE tasks can be transformed into several classification problems, SVM active learning could be used for IE as well. The follows is a general scheme for applying SVM active learning to IE:

1. Randomly choose n_0 documents and manually annotate them as the initial training set S_0. Train the SVM classifiers on S_0 for the IE task.
2. Apply the SVM classifiers to un-annotated documents, and select the n examples with the smallest margins from the un-annotated documents. Label them and add them into the training set.
3. Use the extended training set to re-train the SVM classifiers.
4. Repeat the steps 2 and 3 for a pre-defined number of loops or until the system obtains a pre-defined level of performance.

In the implementation of the above scheme several issues need to be considered. The first one is that which type of example is selected in IE active learning (Step 2), because three types of examples could be used – just one token, a token with its surrounding context and the whole document. We will present experimental results evaluating the three types of examples in our experiments. The margin of a token can be used directly for token examples. On the other hand, selecting documents as examples has to be based on the average confidence of the tokens in the document. In detail, if there are m classifiers in one IE task and for every classifier C_i we select m_0 tokens with the smallest margins $(m_{i1}, \ldots, m_{im_0})$, then we compute the average confidence of the document d as the double sum

$$c_d = \sum_{i=1}^{m} \sum_{j=1}^{m_0} m_{ij} \tag{2}$$

The document with the smallest average confidence would be selected.

The second issue is the optimal setting of parameters in the algorithm. We test different values of the three parameters, n_0 (the number of initial documents for training), n (the number of documents selected in active learning loop), and m_0 (the number of tokens chosen in one document for calculating the average confidence of the document).

The third issue is related to the SVM model. We use the uneven margins SVM in our experiments and obtain the uneven margins SVM classifier by solving a standard SVM problem. However, the margin for each token in the unlabelled documents was computed using the standard SVM classifier. That is because we actually learn a standard SVM and the unlabelled examples which are close to the standard SVM classifier rather than the deduced uneven margins SVM would have important effect to the next round learning in which a standard SVM will be learned as well. As a matter of fact, we did use the margin computed with respect to the uneven margin SVM in active learning and the results were clearly worse than those using margin of the standard SVM, which verified the above arguments. We will discuss the three issues in the experiments presented below.

4 Experiments

4.1 Experimental Datasets and Methodology

We evaluated the approaches of converting multi-class problem into binary classification problems, the uneven margins SVM, and the SVM active learning on three IE benchmarking corpora covering different IE tasks – named entity recognition (CoNLL-2003) and template filling or scenario templates in different domains (Seminars and Jobs). CoNLL-2003[6] is the most recent corpus for named entity recognition. The Seminars and Jobs corpora[7] have been used to evaluate active learning techniques for IE in several papers: [12], [1] and [6].

In detail, we used the English part of the CoNLL-2003 shared task dataset, which consists of 946 documents in the Training Set, 216 document in the Development Set, and 231 documents in the Test Set, all of which are Reuters news articles. The corpus contains four types of named entities — person, location, organisation and miscellaneous.

In the other two corpora domain-specific information was extracted into a number of slots. The Seminars Corpus contains 485 seminar announcements and four slots – start time (stime), end time (etime), speaker and location of the seminar. The Jobs corpus includes 300 software related job postings and 17 slots encoding job details, such as title, salary, recruiter, computer language, application, and platform.

Unless stated otherwise, the experiments in this paper use the parameter settings for the uneven margins SVM, as derived empirically in [27]. Table 1 presents the values of three important parameters: the size of the context window, the SVM kernel type and the uneven margins parameter, used in our experiments for the three corpora, respectively.

Table 1: The values of the three important parameters used in the experiments for the three datasets, respectively. τ refers to the uneven margins parameter for the SVM.

	Context window size	SVM kernel	τ
Conll03	4	quadratic	0.5
Seminars	5	quadratic	0.4
Jobs	3	linear	0.4

4.2 Comparisons of Different Classifier-based Framework for IE

In Section 3.1 we have discussed two frameworks for applying the binary SVM classifier to multi-class problem derived from IE problem. The *BMESO* framework requires to learn classifier(s) for each part of entity and for non-entity token, in which two different approaches were usually used in the implementation. The *one vs. others* approach trains one classifier to separate one class from all other classes. The *pairwise classification* approach learns one classifier for every pair of classes. Another framework, the *BE* framework only trains classifiers for the start and end tokens of entity and the token in a single-token entity is regarded as both the start token and end token of the entity.

We carried out experiments to evaluate these methods on the three corpora. Table 2 presents the results for the three methods, one vs. others, pairwise and the BE framework. We used the standard SVM and the uneven margins SVM for each of the methods, respectively.

[6] See http://cnts.uia.ac.be/conll2003/ner/

[7] See http://www.isi.edu/info-agents/RISE/repository.html.

Table 2: Comparisons of the three methods using the standard SVM and the uneven margins SVM on the three corpora, respectively. The best performance figures for each of the three corpora appear in bold. The 95% confidence intervals for results on the Seminars and Jobs corpora are also presented.

	One vs. others	Pairwise		BE	
	SVM	SVM	SVMUM	SVM	SVMUM
Conll	86.1	85.3	85. 0	85.1	**86.3**
Seminars	81.8±0.6	81.4±0.7	84.2±0.7	80.0±0.4	**84.5±0.6**
Jobs	**80.8±1.4**	80.3±1.0	80.3±1.2	79.0±1.0	**80.8±0.7**

Note that, as we used the same value of uneven margins parameter for all classifiers in one experiment and by doing so the uneven margins SVM had the exactly same results as the standard SVM for the one vs. others method, the table does not present the results of the uneven margins SVM for that method.

For the BE scheme, we used the values of uneven margins parameter listed in Table 1 for each dataset when using uneven margins SVM, because the positive examples were always much less than the negative ones in training set. As for the pairwise method, the situation became a bit complicated as the number of positive training examples may be much less, or much more than, or roughly close to the number of negative examples. Hence, we set τ as the values listed in Table 1 if the positive training examples was at most half of negative ones, set τ as the reciprocal of the values in Table 1 if the negative training examples was at most half of positive ones, and set τ as 1 (corresponding to the standard SVM) in all other cases. For the CoNLL-2003 data we used the training set for learning SVM classifiers and presented results on the test set. For each of the other two corpora, we used half of documents randomly selected for training and other documents for testing, one experiment carried on ten runs and the results were obtained by averaging over the ten runs.

First, if using the standard SVM model, the one vs. others method gave better results than other two methods on all the three corpora [8]. It was significantly better than the BE framework on the two corpora. There is no significant difference between the one vs. others method and the pairwise on the Seminars and Jobs corpora. But the one vs. other method performed clearly better than the pairwise method on the CoNLL-2003 data. So, we think that is why the one vs. others method was mostly used in the applications of the SVM to NLP problems.

However, when using the uneven margins SVM, the results for the BE scheme were significantly improved in the three datasets. For the pairwise method, the results became better on the Seminar data, remained the same for the Jobs data, and was slightly worse on the CoNLL-2003 corpus. As we discussed, the results of the uneven margins SVM were always the same as the standard SVM for the one vs. others method due to its particular mechanism. On the other hand, when using the uneven margins SVM, the BE method obtained better results than the pairwise method and the one vs. other method.

Table 3 presents the computation time and the number of SVM classifiers needed for each method and each corpus, respectively. Note that the SVMUM would take almost the same time as the standard SVM since we obtained the SVMUM by shifting the bias of SVM model. We can see that the BE method needed the least classifiers. The pairwise method required large number of classifiers especially for the Jobs data where there were 17 entity types. Consequently the BE method ran much faster than other two methods.

[8] In contrast, [20] compared the one vs. others method with pairwise method on a variety of machine learning benchmarking datasets and found that the former method performed better than the latter on half of datasets and performed worse on other datasets. Note that none of the datasets they used was from NLP application.

Table 3: Computation time and number of SVM classifiers needed by each of the three methods.

	One vs. others		Pairwise		BE	
	Time	#Classifier	Time	#Classifier	Time	#Classifier
Conll	46308s	17	30400s	136	20544s	8
Seminars	68378s	17	50510s	136	23460s	8
Jobs	45448s	69	70590s	2346	18587s	34

In conclusion, the BE framework with the uneven margins SVM model performed best consistently on the three corpora among all the methods we evaluated. Moreover, it needs least computation time and least computer memory than both the one vs. others method and the pairwise method. Therefore, the BE framework and the uneven margins SVM model is the best combination in the application of SVM to IE problem. We will adopt it in the experiments presented in follows.

4.3 Experiments for Uneven Margins SVM

Named Entity Recognition As already discussed above, we evaluated the uneven margins SVM on the CoNLL-2003 dataset. Since this corpus comes with a development set for tuning the learning algorithm, different settings were tried in order to obtain the best performance Different SVM kernel types, window sizes (namely the number of tokens to the left or right of the token at the centre of window), and the uneven margins parameter τ were tested. We found that quadratic kernel, window size 4 and $\tau = 0.5$ produced best results on the development set. These settings were used in all experiments on the CoNLL-2003 dataset in this paper, unless stated otherwise.

Table 4 presents the results of our system using two learning algorithms, the uneven margins SVM and the standard SVM on the CONLL-2003 test set, together with the results of two participating systems in the CoNLL-2003 shared task: the best system [13] and the SVM-based system [30].

Table 4: Comparison to other systems on CoNLL-2003 corpus: F-measure(%) on each entity type and the overall micro-averaged F-measure. The 95% confidence intervals for results of the two participating systems are also presented. The best performance figures for each entity type and overall appear in bold.

	System	LOC	MISC	ORG	PER	Overall
Our	SVM with uneven margins	89.25	77.79	82.29	90.92	86.30
Systems	Standard SVM	88.86	77.32	80.16	88.93	85.05
Participating	Best one	**91.15**	**80.44**	**84.67**	**93.85**	**88.76**$\pm$**0.7**
Systems	Another SVM	88.77	74.19	79.00	90.67	84.67$\pm$1.0

Our uneven margins SVM system performed significantly better than the participating SVM-based system. However, the two systems are different from each other not only in the SVM models used but also in other aspects such as NLP features and classification framework. Therefore, in order to make a fair comparison between the uneven margins SVM and the standard SVM model, we also present the results of these two algorithms, both using the same framework and the same features. As can be seen from Table 4, under the same experimental settings, the uneven margins SVM still performed better than the standard SVM model.

Template Filling The *Seminar corpus* has been used to evaluate quite a few learning systems. Those include rule learning approaches such as SRV [16], Whisk [36], Rapier [1], BWI [17], SNoW [32] and $(LP)^2$ [5], as well as statistical learning systems such as HMM [14] and maximum entropy (MaxEnt) [4].

The major problem with carrying out comparisons on the seminar corpus is that the different systems used different experimental setups. For instance, SRV, SNoW and MaxEnt reported results averaged over 5 runs. In each run the dataset was randomly divided into two partitions of equal size – one used for training and one for testing. Furthermore, SRV used a randomly selected third of the training set for validation. WHISK's results were from 10-fold cross validation on a randomly selected set of 100 documents. Rapier's and $(LP)^2$'s results were averaged over 10 runs. Finally, BWI's and HMM results were obtained via standard cross validation.

The SVM results reported here are the average over ten runs, following the methodology of Rapier and $(LP)^2$. Table 5 presents the results of our system on seminar corpus, together with the results of the other systems. As far as it was possible, we use the same features as the other systems to enable a more informative comparison. In particular, the results listed in Table 5, including our system, did not use any gazetteer information and named entity recogniser output. The only features in this case are words, capitalisation information, token types, lemmas, and POS tags. The settings for the SVM parameters are shown in Table 1, namely window size 5, quadratic kernel, and $\tau = 0.4$.

We present the F_1 measure for each slot together with the macro-averaged F_1 for the overall performance of the system. Note that the majority of systems evaluated on the seminars and jobs corpora only reported per slot F-measures, without overall results. However, an overall measure is useful when comparing different systems on the same dataset. Hence, we computed the macro-averaged F_1 for the other systems from their per-slot F_1.

Table 5: Comparison to other systems on the Seminar corpus: F_1 (%) on each slot and overall performance (macro-averaged (MA) F_1). The 95% confidential interval for the MA F_1 of our system is also presented. The best results for each slot and the overall performance appear in bold font.

	Speaker	Location	Stime	Etime	MA F_1
SVMUM	69.0	81.3	94.8	92.7	84.5$\pm$0.6
$(LP)^2$	**77.6**	75.0	99.0	95.5	**86.8**
SNoW	73.8	75.2	**99.6**	**96.3**	86.2
MaxEnt	65.3	82.3	**99.6**	94.5	85.4
BWI	67.7	76.7	**99.6**	93.9	84.6
HMM	71.1	**83.9**	99.1	59.5	78.4
Rapier	53.1	73.4	95.9	94.6	79.1
Whisk	18.3	66.6	92.6	86.1	65.7
SRV	56.3	72.2	98.5	77.9	76.0

Table 5 shows that the best results on the different slots are achieved by different systems and that the best overall performance is achieved by $(LP)^2$. While the uneven margins SVM did not achieve the best performance, it still outperformed many other systems. Also, it is worth noting that our system using SVM active learning with uneven margins outperforms $(LP)^2$ on the Seminars corpus, as shown in Table 10 in Section 4.4.

On the *Jobs corpus* our system is compared to several state-of-the-art learning systems, including the rule based systems Rapier [1], $(LP)^2$ [5] and BWI [17], the statistical system HMM [17], and the double classification system [35]. As before, in order to make the comparison as informative as possible, the same settings were adopted in our experiments

as those used by $(LP)^2$, which previously reported the highest results on this dataset. In particular, the results are obtained by averaging the performance in ten runs, using a random half of the corpus for training and the rest for testing. Only basic NLP features are used: token form, capitalisation information, token types, and lemmas.

Table 6: Comparison to other systems on the Jobs corpus: F_1 (%) on each entity type and overall performance as macro-averaged (MA) F_1. The 95% confidential interval for the MA F_1 of our system is also presented. The highest score on each slot and overall performance appears in bold.

Slot	SVMUM	$(LP)^2$	Rapier	DCs	BWI	HMM	semi-CRF
Id	97.7	**100**	97.5	97	100	–	–
Title	49.6	43.9	40.5	35	50.1	**57.7**	40.2
Company	77.2	71.9	70.0	38	**78.2**	50.4	60.9
Salary	**86.5**	62.8	67.4	67	–	–	–
Recruiter	78.4	**80.6**	68.4	55	–	–	–
State	92.8	84.7	90.2	**94**	–	–	–
City	**95.5**	93.0	90.4	91	–	–	–
Country	**96.2**	81.0	93.2	92	–	–	–
Language	86.9	**91.0**	81.8	33	–	–	–
Platform	80.1	**80.5**	72.5	36	–	–	–
Application	70.2	**78.4**	69.3	30	–	–	–
Area	46.8	**53.7**	42.4	17	–	–	–
Req-years-e	**80.8**	68.8	67.2	76	–	–	–
Des-years-e	81.9	60.4	**87.5**	47	–	–	–
Req-degree	**87.5**	84.7	81.5	45	–	–	–
Des-degree	59.2	65.1	**72.2**	33	–	–	–
Post date	99.2	**99.5**	**99.5**	98	–	–	–
MA F_1	**80.8±0.7**	77.2	76.0	57.9	–	–	–

Preliminary experiments established that the SVM with linear kernel obtained better results than SVM with quadratic kernel on the Jobs corpus [27]. Hence we used the SVM with linear kernel in the experiments on the Jobs data.

Table 6 presents the results of our uneven margins SVM system as well as the other six systems which have been evaluated on the Jobs corpus. Note that the results on all 17 slots are available only for three previous systems: Rapier, $(LP)^2$ and double classification. We computed the macro-averaged F_1 (the mean of the F_1 of all slots) for our system as well as for the three fully evaluated systems in order to compare the overall performance.

The results show that the overall performance of the uneven margins SVM is significantly better than the other three fully evaluated systems. The double classification system had much worse overall performance than our system and the other two fully evaluated systems. HMM was evaluated only on two slots. It achieved the best result on one slot but had a significantly worse performance on the other slot. BWI obtained better results than ours on three slots. Unfortunately, BWI results for the other slots are not available, thus making it impossible to compare the two algorithms on the entire Jobs dataset.

Effect of the Uneven Margins Parameter. A number of experiments were conducted to investigate the influence of the uneven margins parameter on the SVM performance. Table 7 shows the results with several different values of the uneven margins parameter on the three datasets – CoNLL-2003, Seminars and Jobs. The SVM with uneven margins ($\tau < 1.0$) had better results than the standard SVM ($\tau = 1$). We can also see that the results were similar for τ between 0.6 and 0.4, showing that the results are not particularly

Table 7: The effects of uneven margins parameter of the SVM: macro averaged F_1(%) on the the three datasets CoNLL-2003 (test set), Seminars and Jobs. The 95% confidential intervals for the Seminars and Jobs datasets are also presented, showing the statistical significances of the results. In bold are the best performance figures for each dataset.

τ	1.0	0.8	0.6	0.4	0.2
Conll	85.0	86.0	**86.2**	85.9	81.6
Seminars	80.0±0.4	82.3±0.5	84.1±0.6	**84.5±0.6**	80.9±0.7
Jobs	79.0±1.0	79.9±0.9	**81.0±0.6**	80.8±0.7	79.0±0.9

sensitive to the value of the uneven margins parameter.

Our conjecture was that the uneven margins parameter was even more helpful on smaller training sets, because the smaller a training set is, the more imbalanced it could be. Therefore we carried out experiments on a small numbers of training documents. Table 8 shows the results of the standard SVM and the uneven margins SVM on different numbers of training documents from CoNLL-2003, Seminars and Jobs datasets, respectively. The performance of both SVMs improves consistently as more training documents are used. Moreover, compared to the results on the full training sets (see Table 7), the smaller the training set is, the better the results of the uneven margins SVM are in comparison to the standard SVM.

Table 8: The performances of the SVM system with small training sets: macro-averaged F_1(%) on the three datasets CoNLL-2003 (test set), Seminars and Jobs. The uneven margins SVM ($\tau = 0.4$) is compared to the standard SVM model with even margins ($\tau = 1$). The 95% confidential intervals for the Seminars and Jobs datasets are also presented, showing the statistical significances of the results.

size	10	20	30	40	50
	$\tau = 0.4$				
Conll	60.5	66.6	70.7	72.2	72.3
Seminars	58.1±3.4	67.1±2.8	73.6±1.5	76.5±1.8	78.2±1.6
Jobs	51.6±1.9	60.9±1.8	65.7±1.4	68.6±1.4	71.1±1.8
	$\tau = 1$				
Conll	51.8	60.2	66.0	67.4	68.9
Seminars	41.5±3.4	53.1±2.5	60.9±1.6	65.3±2.1	68.9±1.5
Jobs	47.1±2.4	56.5±2.2	61.4±1.9	65.4±1.4	68.1±1.5

4.4 Experiments with SVM Active Learning

As already discussed in Section 3.3, active learning first selects some examples for initial learning. Then in each learning round, according to the active learning algorithm, some more examples are selected for training.

For the CoNLL-2003 corpus, the initial training documents were chosen randomly from the Training Set and in each active learning round further examples were selected from the remaining documents in the Training Set. The results reported in this paper are on the Test Set.

For each of the other two corpora (Seminars and Jobs), the initial training set was chosen randomly from the whole corpus and each active learning loop selected samples from the remaining documents. Then all documents not used for training were used for testing. All

results reported below are the average from ten runs.

The first experiments below use entire documents as samples. The other two types of samples – token and token with context – are discussed at the end part of this section.

Active Learning vs. Random Selection. We first compare SVM active learning with a random selection baseline. For all three datasets (CoNLL-2003, Seminars and Jobs), two documents were chosen as the initial training set and then each active learning loop selected the document with the smallest average confidence as the source for additional training data. The average confidence of a document was calculated using the confidence for 5 tokens on both CoNLL-2003 and Seminars corpora and 2 tokens on the Jobs corpus (see below discussions about how these values were determined).

Figure 2 presents the learning curves for active learning and random selection on the three datasets. Both used the uneven margins SMV. The results for the standard SVM with random selection are also presented at some data points for comparison.

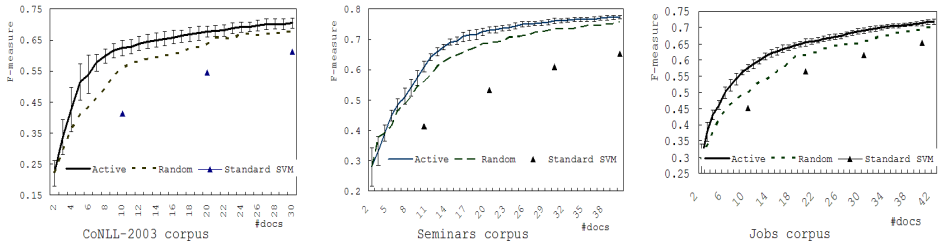

Figure 2: Learning curves for overall F1 on the three datasets. Active learning is compared with random selection for the uneven margins SVM. The results for the standard SVM model and random selection are also presented for comparison. The error bars show 95% confidence intervals. For clarity, only the error bars for active learning are plotted.

First, as expected, the results of active learning are clearly better than those of random selection on all three datasets. It is also worth noting that again the uneven margins SVM performs significantly better than the standard SVM with randomly selection.

Secondly, the gap between active learning and random selection widens after the first few learning loops, however, the difference is dataset-dependent, i.e., smaller on some sets than on others. The learning curves become flat after approximately 20 loops for both active learning and random selection.

Thirdly, for clarity, only the confidence intervals for active learning are plotted in Figure 2. As a matter of fact, the confidence intervals for random selection are bigger than the corresponding ones for active learning at almost all data points for all three datasets, showing that the results of active learning are more stable.

Finally, we can compare our results with those in the previous works. [12] also used the Seminars corpus to evaluate several active learning techniques based on the learning algorithm $(LP)^2$, and adopted similar experimental settings to ours. Table 9 compares our results with the results presented in [12][9].

We can see that our active learning had much better performance than the best results in [12]. Actually the results of our active learning are close to the optimal results presented there, which was an estimate of the upper bound on the performance of any selection strategy.

[9]Because [12] presented their results in the form of graphs instead of tables, their results listed in Table 9 were estimated by us from their graphs.

Table 9: Comparisons of our results with the best results and the upper bounds presented in [12]: the overall F_1 on the Seminar corpus.

#docs	SVMUM+AL	Finn03	Upper bound
20	0.730	0.61	0.77
30	0.761	0.66	0.79

Table 10 compares our results with a state of the art system for adaptive IE, Amilcare based on the $(LP)^2$ algorithm, on the Seminar corpus [7]. We can see that the uneven margins SVM with random selection performed worse than Amilcare, but that in turn was outperformed by the uneven margins SVM with active learning.

Our results on the Jobs dataset are better than those achieved by another active learning method based on the IE system Rapier [1]. For example, with 40 training documents, the overall F_1 of the SVM active learning algorithm is 0.71, higher than Rapier's corresponding figure (around 0.64).

Table 10: Comparison of our algorithms with Amilcare: F_1 (%) for each slot of the Seminars corpus and the macro averaged F_1 for overall performance. AL+SVMUM refers to the SVM active learning with uneven margins SVM model. SVMUM refers to the uneven margins SVM using random selection. The highest score on each slot and overall performance appears in bold.

	#docs	SVMUM+AL	SVMUM	Amilcare
stime	30	**90.5**	88.9	84.0
etime	20	**89.6**	80.6	82.3
speaker	25	**53.3**	50.5	50.6
location	30	68.9	63.7	**70.0**
MA F_1		**75.6**	70.9	71.7

Three Parameters in Active Learning. As discussed in Section 3.3, three parameters impact the performance of SVM active learning. The first one is the *number of tokens* used for computing the average confidence of a document, namely m_0 in Equation (2). As there may be some outliers in a classification problem, using one token could lead to an unstable performance. On the other hand, if many tokens are used, the tokens with large margins would overwhelm the informative tokens with smaller margins. Table 11 presents the results with different values of m_0 for the Seminars and Jobs datasets. We can see that too small or too large value of m_0 produces worse results than using a value 3 or 5.

Table 11: Different number of tokens used for computing the average confidence of document: macro averaged F_1 (%) with the 95% confidence interval for the two datasets Seminars and Jobs.

$m_0 =$	1	3	5	10
Seminars	66.2±1.6	70.4±1.4	**72.0±1.8**	69.3±1.7
Jobs	64.2±1.6	**65.2±1.2**	65.0±0.7	63.1±1.7

The second parameter is the *number of initial documents* randomly chosen for training, namely n_0, as introduced in Section 3.3. On the one hand, the smaller n_0 is, the earlier we can take advantage of active learning. On the other hand, since active learning uses the current model to choose the next examples, too few initial examples could result in a bad SVM model which in turn would be harmful to the quality of the selected examples. Table 12 shows the results for n_0=1, 2, 4, and 10, respectively. n_0=2 obtained the best result

for Seminars and n_0=4 for the Jobs corpus. However, there seems to be no significant difference between the results on the Jobs dataset.

Table 12: Different number of initial training document for SVM active learning: macro averaged F_1 (%) with the 95% confidence interval for the two datasets Seminars and Jobs.

n_0=	1	2	4	10
Seminars	72.2±0.7	**73.0 ±0.9**	72.0±1.8	68.3±1.6
Jobs	64.1±1.2	65.0±1.1	**65.8±0.8**	65.2±1.2

The last parameter discussed here is the *number of documents* n selected in each active learning loop. If each loop chooses only one document prior to re-training the model, then the results will be more accurate than those obtained by selecting two or more documents each time. On the other hand, the more documents are selected in one loop, the fewer loops the system may need in order to obtain a given performance. In other words, if the second best example is not much less informative than the best one in one active learning loop, then we may use the second best example as well as the first one, in order to save computation time. Table 13 shows that n=1 gave slightly better results than n = 2 or 3 for both datasets, but the computation time for n=2 was half of that for n=1 while still achieving similar performance.

Table 13: Different number of documents selected in one active learning loop: macro averaged F_1 (%) with the 95% confidence interval for the two datasets Seminars and Jobs.

n=	1	2	3
Seminars	**73.0±1.0**	72.8±0.9	71.5±1.2
Jobs	**65.0±1.0**	64.2±0.7	64.1±1.0

Three Types of Samples. When SVM active learning is applied to IE it can select one or more documents in each loop based on document's average confidence. Alternatively, active learning can select only the most informative *tokens* from the unlabelled documents or a fragment of text containing the most valuable token and the surrounding tokens as a sample.

If a full document is selected in each loop, the user must decide whether or not every token in the document belongs to each type of entity considered. If just the token or text fragment is selected for one classifier, then the user would only decide whether the selected token or each token in the fragment is the start (or end) token of the entity which type is specified by classifier. Therefore, compared to full document samples, token or text fragment sampling could save a great deal of manual annotation time in each active learning loop. Therefore we investigated how the performance of token and text fragment sampling compares to full document sampling.

Figure 3 plots the learning curves for using a token and a token with context as the sample, respectively. The learning curve for full document samples is also presented for comparison. Two initial training documents were used in all experiments. We selected 5 or 2 tokens respectively for Seminars and Jobs datasets in each learning loop for both token and token with context samples. The first 40 active learning loops are plotted for all learning curves.

Not surprisingly, the results using document samples are the highest and token with context performed better than token alone in most cases. However, it is worth noting that the performance of token with context as sample achieved similar performance to full document sampling in the first few loops. The learning curves for token samples become flatter in the late stage of learning, compared to the curve for full document samples. The learning curve for token with context is even worse — it decreases after some learning loops.

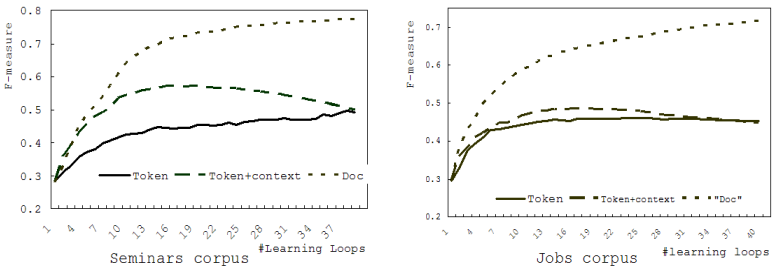

Figure 3: Learning curves for active learning using three types of sample on the two datasets, respectively.

On the other hand, while the user has to annotate hundreds of tokens in full document sampling, for token or token with context sampling they just need to decide on just a few tokens. For example, in the experiments on the Seminars corpus presented in Figure 3, in each learning loop the user has to annotate 337 tokens on average for full documents, or decide whether each of 5 or 55 tokens belong to the entity of particular type respectively for the token and the text fragment. The annotation time for document sampling would be about 67 and 6 times more than that for token and token with context sampling, respectively.

5 Related Work

SVM Learning for NLP. SVM has been applied successfully to a variety of NLP problems. The methodology of applying SVM to IE presented in this paper was used in most applications, i.e., transform the NLP task into a number of binary classification problems and train a classifier for each. A very high dimensional binary feature vector is often constructed for each token using the NLP features from the token as well as some surrounding tokens.

On some NLP problems SVM with quadratic kernel performs better than linear or other types of kernel. On the other hand, as linear kernel SVM needs much less training and application time than quadratic kernel SVM, some systems use a linear kernel and meanwhile encode some co-occurrences of features into the feature vector explicitly, which is equivalent to using some kind of semi-quadratic kernel (see e.g. [22]). This kind of implementation enables the SVM with linear kernel to achieve as good performance as with quadratic kernel but it is much faster than using quadratic kernel. The SVM performance on NLP problem is often robust to other SVM parameters.

Once the SVM classifiers are learned for one NLP problem, we have to combine the results from the binary SVM classifiers into a solution of the original NLP problem. Most previous works adopted the one vs. others method to learn an SVM classifier for each part of entity and for non-entity as well. They assigned an instance the class which SVM classifier had the maximal output on the instance. For example, the one vs. others method was used in [22] for POS tagging, [11] for WSD, [45] for relation extraction and [19] for semantic role labeling. However, some works made some modifications on the process. For example a Viterbi search process was used for choosing the best combination of tags for a sequence of tokens respectively in [21] for named entity recognition and in [23] for Japanese dependency analysis.

There were a few works dealing with imbalanced data in NLP applications. [22] constructs training data from a dictionary extracted from the training corpus rather than training on

the corpus itself. This eliminated many negative examples and is similar to the under-sampling method. [24] used *pairwise classification* which trains a classifier for each pair of classes. In pairwise classification the negative and positive examples are respectively from two classes, so the training data would be less imbalanced in most cases than the one vs. others method.

SVM active learning. [2] proposed the margin-based approach to SVM active learning and used it for USPS hand-written digit recognition. They found that SVM active learning performed much better than SVM with random selection of examples. In the meantime, [34] studied a similar method for SVM active learning. They applied the method to text categorisation and obtained significantly better results than using random selection. [38] studied the same algorithms for SVM active learning as those in [34] in similar settings. They compared SVM active learning with two committee-based active learning algorithms and found that the SVM active learning approach obtained the best results. SVM active learning was also successfully used for image retrieval [37], computer aided drug design [42] and spoken language understanding [40].

There are several papers studying SVM active learning for NLP problems. [33] investigated SVM active learning for Japanese word segmentation. He had the same basic findings as ours, namely that SVM active learning can significantly improve performance. He also found that a small unlabelled data pool was helpful in the early stages of SVM active learning and proposed algorithms for determining the appropriate size of this data pool during learning. We believe that his algorithms would also be helpful for information extraction. On the other hand, he did not investigate other settings in SVM active learning, such as different types of examples and number of initial training documents, which are studied in this paper.

The thesis [41] studied SVM active learning for named entity recognition and NP chunking. Similar to [38], Vlachos studied different approaches for selecting unlabelled examples based on the confidence score of the SVM classifier. This work also used the CoNLL-2003 share task data, which enables direct comparisons between his results and ours. Figure 5.5 in the thesis presented the same kind of learning curves as those in Figure 2 of this paper. Both figures show that active learning has significantly better performance than passive learning. On the other hand, our F-measures for both random learning and active learning are much higher than the corresponding figures in the thesis. The difference may be due to the different settings between his experiments and ours, such as NLP features, framework, and SVM models. It is worth noting the all previous works on SVM active learning used the standard SVM model while our work explored the uneven margins SVM.

6 Conclusions

SVM is a popular learning algorithm for solving NLP tasks. We investigated two advanced techniques for helping SVM deal with imbalanced training data and the difficulty of obtaining human-annotated examples – two problems that frequently arise in NLP datasets.

We present a new approach towards dealing with imbalanced training data by introducing the uneven margins parameter to the SVM model. We also investigated SVM active learning and the different strategies that can be used with it – full document, single token, and tokens in context. We also tested uneven margins SVM and SVM active learning together on several IE tasks, including named entity recognition and template filling. The uneven margins SVM obtained better experimental results than the standard SVM model. SVM active learning also achieved better performance than conventional learning methods. Moreover, when the two are combined together, we achieve the best results on the Seminars

corpus, a benchmark dataset for IE. The uneven margins SVM also obtains the best results on the Jobs corpus, another IE benchmark dataset.

For the SVM model with bias term, the uneven margins SVM model can be obtained from a related standard SVM classifier by changing the bias term. Hence the uneven margins SVM can be easily solved using any publicly available SVM package. On the other hand, it performs much better than the standard SVM on imbalanced training data. We believe that uneven margins SVM could lead to a similar performance on NLP tasks other than IE. Similarly, SVM active learning for IE presented in this paper is general so that it can also be used on other NLP problems, where similarly good performance is expected.

We also evaluated the approaches of converting multi-class classification problem into binary classification problems, which is required when using the binary SVM classifier for the NLP applications. Our experiments showed that the combination of the BE framework with the uneven margins SVM is the best option for applying the SVM to the IE problem, as they gave the best results among all the combinations and the BE framework also needed much less computation time and computer memory than other methods. In comparison with the one vs. others method and the pairwise method, the BE method was rarely used in the previous applications of the SVM in NLP, mainly because it gave worse results when using the standard SVM model. However, the uneven margins SVM significantly improved the results over the SVM in the BE framework for the IE task. We expect that the uneven margins SVM with the BE framework would give good results on many other NLP applications as well.

Acknowledgements: This research is supported by the EU-funded SEKT project (http://www.sekt-project.com).

References

[1] M. E. Califf. *Relational Learning Techniques for Natural Language Information Extraction*. PhD thesis, University of Texas at Austin, 1998.

[2] C. Campbell, N. Cristianini, and A. Smola. Query Learning with Large Margin Classiffiers. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, 2000.

[3] X. Carreras, L. Màrquez, and L. Padró. Learning a perceptron-based named entity chunker via online recognition feedback. In *Proceedings of CoNLL-2003*, pages 156–159. Edmonton, Canada, 2003.

[4] H. L. Chieu and H. T. Ng. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 786–791, 2002.

[5] F. Ciravegna. $(\mathrm{LP})^2$, an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, 2001.

[6] F. Ciravegna, A. Dingli, D. Petrelli, and Y. Wilks. User-System Cooperation in Document Annotation Based on Information Extraction. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 122–137, Siguenza, Spain, 2002.

[7] F. Ciravegna and Y. Wilks. Designing Adaptive Information Extraction for the Semantic Web in Amilcare. In S. Handschuh and S. Staab, editors, *Annotation for the Semantic Web*. IOS Press, Amsterdam, 2003.

[8] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[9] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[10] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[11] G. Escuderoz, L. Marquezx, and G. Rigau. TALP System for the English Lexical Sample Task. In *SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

[12] A. Finn and N. Kushmerick. Active learning selection strategies for information extraction, 2003.

[13] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named Entity Recognition through Classifier Combination. In *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada, 2003.

[14] D. Freigtag and A. K. McCallum. Information Extraction with HMMs and Shrinkage. In *Proceesings of Workshop on Machine Learnig for Information Extraction*, pages 31–36, 1999.

[15] D. Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Carnegie Mellon University, 1998.

[16] D. Freitag. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, 39(2/3):169–202, 2000.

[17] D. Freitag and N. Kushmerick. Boosted Wrapper Induction. In *Proceedings of AAAI 2000*, 2000.

[18] A. M. Gliozzo, C. Giuliano, and R. Rinaldi. Instance Filtering for Entity Recognition. *SIGKDD Explorations*, 2005. Special Issue: Text Mining and Natural Language Processing.

[19] K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. Semantic Role Labeling by Tagging Syntactic Chunks. In *Proceedings of CoNLL-2004*, pages 110–113, 2004.

[20] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 2002.

[21] H. Isozaki and H. Kazawa. Efficient Support Vector Classifiers for Named Entity Recognition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 390–396, Taipei, Taiwan, 2002.

[22] J.Gimenez and L. Marquez. Fast and Accurate Part-of-Speech Tagging: The SVM Approach Revisited. In *Proceedings of the International Conference RANLP-2003 (Recent Advances in Natural Language Processing)*, pages 158–165. John Benjamins Publishers, 2003.

[23] T. Kudo and Y. Matsumoto. Use of Support Vector Learning for Chunk Identification. In *Proceedings of Sixth Conference on Computational Natural Language Learning (CoNLL-2000)*, 2000.

[24] T. Kudoh and Y. Matsumoto. Japanese Dependency Structure Analysis Based on Support Vector Machines. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 2000.

[25] Y. Lee, H. Ng, and T. Chia. Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. In *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 137–140, 2004.

[26] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397, 2004.

[27] Y. Li, K. Bontcheva, and H. Cunningham. SVM Based Learning System For Information Extraction. In *Proceedings of Sheffield Machine Learning Workshop*, Lecture Notes in Computer Science. Springer Verlag, 2005.

[28] Y. Li, C. Miao, K. Bontcheva, and H. Cunningham. Perceptron Learning for Chinese Word Segmentation. In *Proceedings of Fourth SIGHAN Workshop on Chinese Language processing (Sighan-05)*, pages 154–157, Korea, 2005.

[29] Y. Li and J. Shawe-Taylor. The SVM with Uneven Margins and Chinese Document Categorization. In *Proceedings of The 17th Pacific Asia Conference on Language, Information and Computation (PACLIC17)*, Singapore, Oct. 2003.

[30] J. Mayfield, P. McNamee, and C. Piatko. Named Entity Recognition Using Hundreds of Thousands of Features. In *Proceedings of CoNLL-2003*, pages 184–187. Edmonton, Canada, 2003.

[31] T. Nakagawa, T. Kudoh, and Y. Matsumoto. Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, 2001.

[32] D. Roth and W. T. Yih. Relational Learning via Propositional Algorithms: An Information Extraction Case Study. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1257–1263, 2001.

[33] M. Sassano. An Empirical Study of Active Learning with Support Vector Machines for Japanese Word Segmentation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[34] G. Schohn and D. Cohn. Less is More: Active Learning with Support Vector Machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, 2000.

[35] A. De Sitter and W. Daelemans. Information extraction via double classification. In *Proceedings of ECML/PRDD 2003 Workshop on Adaptive Text Extraction and Mining (ATEM 2003)*, Cavtat-Dubrovnik, Croatia, 2003.

[36] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272, 1999.

[37] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proc. of ACM Int. Conf. on Multimedia*, pages 107–118, 2001.

[38] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[39] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proceedings of the 21 st International Conference on Machine Learning*, Banff, Canada, 2004.

[40] G. Tur, R.E. Schapire, and D. Hakkani-Tur. Active Learning for Spoken Language Understanding. In *Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 276–279, 2003.

[41] A. Vlachos. Active learning with support vector machines. MSc thesis, University of Edinburgh, 2004.

[42] M. Warmuth, G. Ratsch, M. Mathieson, J. Liao, and C. Lemmen. Active learning in the drug discovery process. In *Advances in Neural Information Processing Systems 15 (NIPS-2002)*, pages 1449–1456, 2002.

[43] H. Yamada and Y. Matsumoto. Statistical Dependency Analysis with Support Vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*, 2003.

[44] J. Zhang and I. Mani. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.

[45] G. Zhou, J. Su, J. Zhang, and M. Zhang. Exploring Various Knowledge in Relation Extraction. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 427–434, 2005.