Adapting SVM for Data Sparseness and Imbalance: A Case Study on Information Extraction

Yaoyong Li, Kalina Bontcheva, Hamish Cunningham

Department of Computer Science, The University of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK.
E-mail: {yaoyong,kalina,hamish}@dcs.shef.ac.uk

(Received 29 January 2008)

Abstract

Support Vector Machines (SVM) have been used successfully in many Natural Language Processing (NLP) tasks. The novel contribution of this paper is in investigating two techniques for making SVM more suitable for language learning tasks. Firstly, we propose an SVM with uneven margins (SVMUM) model to deal with the problem of imbalanced training data. Secondly, SVM active learning is employed in order to alleviate the difficulty in obtaining labelled training data. The algorithms are presented and evaluated on several Information Extraction (IE) tasks, where they achieved better performance than the standard SVM and SVM with passive learning, respectively. Moreover, by combining SVMUM with the active learning algorithm, we achieve the best reported results on the seminars and jobs corpora, which are benchmark datasets used for evaluation and comparison of machine learning algorithms for IE. In comparison to previous methods dealing with the same problems, our methods are both effective and efficient, which is valuable features for real-world applications. Due to the similarity in the formulation of the learning problem for IE and for other NLP tasks, the two techniques are likely to be beneficial in a wide range of applications.
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 (J.Gimenez and Marquez2003; Nakagawa, Kudoh, and Matsumoto2001), word sense disambiguation (Lee, Ng, and Chia2004), NP (noun phrase) chunking (Kudo and Matsumoto2000), information extraction(Isozaki and Kazawa2002; Li, Bontcheva, and Cunningham2005a), relation extraction (Zhou et al.2005), semantic role labeling (Hacioglu et al.2004), and dependency analysis (Kudoh and Matsumoto2000; Yamada and Matsumoto2003). 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 the solution to the original NLP problem.

When compared to other classification problems, NLP classification tasks have several unique characteristics, which should be taken into consideration, when applying machine learning algorithms. 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.

\(^1\) Support vector machines have also been formulated as a multi-class classifier in various forms (see e.g. (Crammer and Singer2001)), and obtained encouraging results for some applications including several NLP problems (see e.g. (Tsochantaridis et al.2004)), where multi-class SVM obtained better results than the binary model. However, although multi-class SVMs are a promising research area, their implementation is more complicated than the binary one. Moreover, (Hsu and Lin2002) compared the binary SVM with one form of multi-class SVM presented in (Crammer and Singer2001) on several standard machine learning datasets and their results showed no clear difference in performance between binary and multi-class SVM. Therefore, this paper only considers the binary SVM classifier. But certainly investigation of the applications of multi-class SVM in the NLP problems is a worth-doing work.
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.

Therefore, when SVMs are applied to NLP tasks, these particular aspects should be taken into account in order to obtain a practical system with good performance. The novel contribution of this paper is in investigating two techniques for making SVM more suitable for language learning tasks. Firstly, we propose an SVM with uneven margins (SVMUM) model to deal with the problem of imbalanced training data. Secondly, SVM active learning is employed in order to alleviate the difficulty in obtaining labelled 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.

In comparison with other methods which transform the training data (e.g. sub-sampling or oversampling, see Section 3), our method adapts the learning algorithm directly to deal with imbalanced data. Our method is more effective. On the other hand, in comparison with other modified SVM such as multi-class SVM for imbalanced data and transductive SVM for exploiting unlabelled data (see e.g. (Collobert et al.2006)), our SVMUM and active learning are simple and efficient, because our methods just need solve the standard binary SVM and use some simple transformations, while both multi-class SVM and transductive SVM need implement and solve some new and more complicated optimalisation problems. Hence our methods are both effective and efficient, which are the desirable features for some real-world applications which may have hundreds of classes and millions of instances.

The rest of the paper is structured as follows. Section 2 discusses the IE tasks and the SVM learning, and presents the way in which SVM classifiers are used to solve IE tasks. Section 3 focuses on imbalanced data and the SVMUM algorithm, whereas Section 4 discusses SVM active learning for IE. Section 5 evaluates the algorithms on three benchmark corpora for IE, with a particular emphasis on measuring the
usefulness of active learning. In the end, section 6 discusses related work and Section 7 summarises our findings.

2 SVM Learning for IE

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. For example, 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 business intelligence, 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 induces a set of rules from training examples. There are many such rule-based learning systems, e.g. SRV (Freitag1998), RAPIER (Califf1998), WHISK (Soderland1999), BWI (Freitag and Kushmerick2000), and \((LP)^2\) (Ciravegna2001). Statistical systems learn a statistical model or classifiers, such as HMMs (Freitag and McCallum1999), Maximal Entropy (Chieu and Ng.2002a), SVM (Isozaki and Kazawa2002; Mayfield, McNamee, and Piatko2003; Li, Bontcheva, and Cunningham2005a), and Perceptron (Carreras, Márquez, and Padró2003; Li, Bontcheva, and Cunningham2005b).

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.
2.1 SVM Learning For NLP

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 training examples (except a small number of examples as outliers) (see e.g. (Cristianini and Shawe-Taylor2000)). We can see the margin of training examples to the classification hyperplane as a measure of tolerance of the classification model to discrepancy between training examples and testing examples. The bigger the margin is, the more tolerance the model may have for the discrepancy. Therefore on classification tasks 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, because the former maximises the tolerance for the discrepancy between training and testing examples. There are theoretical results specifying lower bound of generalisation capability from the maximum margin learning algorithms like SVM (see (Shawe-Taylor and Cristianini1999; Cristianini and Shawe-Taylor2000)), which also indicates good generalisation capability of the SVM algorithm. Another SVM feature is that, by using different types of kernel function, it can explore different 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.

Specifically in the case of NLP tasks, instances are typically represented by very high dimensional but sparse feature vectors, which may lead to positive and negative examples being distributed into two distinctly separate areas of the feature space. This is particularly helpful for SVM’s search for a classification hyperplane and also for its generalisation capability. In fact, this is the main reason why SVMs can achieve very good results on a variety of NLP tasks. Such very high dimensional representation is achieved by forming the feature vector explicitly from text using a large number of linguistic features and in many cases by exploring the so-called kernel function to map the feature vector into even higher dimensional space.
Furthermore, as SVM is an optimal margin classifier, the distance of an example to the SVM classification hyperplane indicates how important is the example to the SVM learning. The examples being close to the SVM hyperplane are crucial for the learning. Consequently, SVM active learning is based on the distance of unlabelled examples to the SVM hyperplane (see Section 4 for detailed explanations).

2.2 SVM-Based Information Extraction

The SVM-based IE approach adopted in this work 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.

The aim of the pre-processing is to form input vectors from documents. Each document is first processed using the open-source ANNIE system, which is a part of the GATE NLP toolset\(^2\) (Cunningham et al. 2002). This produces a number of linguistic features, including capitalisation information, token kind (i.e., word, number, punctuation), lemma, part-of-speech (POS) tag, semantic classes from gazetteers, and named entity types according to ANNIE's rule-based recogniser.

Based on this linguistic information, an input 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. Since in IE the context of the token is usually as important as the token itself, the features in the input vector come not only from the current token, but also from preceding and following ones. 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 use is the \textit{reciprocal scheme}, which weights the surrounding tokens reciprocally to the distance to the token in the centre of the context window. This reflects

\(^2\) Available from http://www.gate.ac.uk/
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 of features (Li, Bontcheva, and Cunningham2005a).

The key part of the approach is to convert the recognition of information entities into binary classification tasks – one to decide whether a token is the start of an entity and another one for the end token.

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, the document is scanned from left to right 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.

3 Imbalanced Training Data and SVMUM

As already discussed in Section 1, NLP classification problems usually have 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. One has to use small training data for learning in some applications such as mixed-initiative text annotations (Day et al.1997) and adaptive IE (Ciravegna et al.2002). Two approaches have been studied so far to deal with imbalanced data for IE
Adapting SVM for Data Sparseness and Imbalance

tasks. The first one under-samples the majority class or over-samples the minority class in order to obtain a relatively balanced training data (Zhang and Mani2003). 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 (Carreras, Márquez, and Padró2003; Sitter and Daelemans2003). 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.

A different approach to handling the imbalanced data in IE is investigated here: namely, modifying the SVM learning algorithm for balanced classification to deal better with imbalanced data.

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, then a maximal margin classifier (such as SVM) learned from an unrepresentative training set may have poor generalisation performance, as illustrated in Figure 1.

Figure 1 shows a simple 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
Fig. 1. An illustrative 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.

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. (Lewis et al. 2004; Li and Shawe-Taylor 2003)).

However, as can be seen in Figure 1, if the classification hyperplane could be moved away from the positive training examples in the imbalanced dataset, then the classifier would classify more unseen data correctly, i.e., it would have better generalisation performance. Therefore, 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 $\tau$ into the SVM optimisation problem to control the ratio of the positive margin over the negative margin (for details see (Li and Shawe-Taylor 2003)).

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 $y_i (= +1 \text{ or } -1)$ its label, SVMUM is obtained by solving the quadratic optimisation problem:

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

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

where a parameter $\tau$ was added to the constraints of the optimisation problem for the standard SVM formation. $\tau$ 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, as argued above, for imbalanced training data, a larger positive margin than negative one (namely $\tau < 1$) would be beneficial for the generalization capability of the SVM classifier.

When applying SVMUM to a problem, we first have to determine a value for the uneven margins parameter $\tau$. 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 $\tau$ 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 $\tau$ can be empirically determined, for example, by n-fold
cross-validation on the training set or using a hold-out development set. On the other hand, the experimental results presented in (Li, Bontcheva, and Cunningham2005a) show that the performance of SVMUM 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 $\tau$ is able to help the SVMUM to achieve significantly better results than the standard SVM model (see Section 5).

As showed in (Li and Shawe-Taylor2003), for the SVM model with bias term, the solution of SVMUM can be obtained from a related standard SVM via a transformation:

\begin{align}
  w_2^* &= w_1^* \quad (1) \\
  b_2^* &= b_1^* + \frac{1 - \tau}{1 + \tau} \quad (2)
\end{align}

where $\{w_1^*, b_1^*\}$ is the SVM model for the standard SVM with cost factor $C$ for some training data, and $\{w_2^*, b_2^*\}$ is the SVM model for the SVM with uneven margin $\tau$ and cost factor $C_\tau = (1 + \tau)/(2 \times C)$ for the same training data (for details see (Li and Shawe-Taylor2003)). The transformation is simple — basically it amounts to adding a $\tau$-related term to the bias term $b$ of the corresponding standard SVM model. Therefore, in order to achieve computational gains, the SVMUM 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\(^3\), and then the solution of SVMUM is obtained through a transformation.

On the other hand, the transformation means that the SVMUM 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

\(^3\) The SVM\(^\text{light}\) package, available from http://svmlight.joachims.org/, was used to learn the SVM classifiers in our experiments.
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.

3.1 Comparing SVMUM against other SVM approaches that deal with
imbalanced data

It has been demonstrated previously that SVM learning obtains poor results on im-
balanced data where the number of positive and negative examples are very different
(see e.g. (Lewis et al.2004)). Several techniques have been proposed to alleviate this
problem. The first one was presented in (Morik, Brockhausen, and Joachims1999),
which differentiated the cost factor ($C$) in the SVM’s optimal problem between the
positive training examples (as $C_+$) and the negative ones (as $C_-$). It often com-
pensated the smaller number of positive (or negative) examples by setting a higher
cost value for them. This approach was implemented by the SVM package SVMlight
((Joachims1999a)) in which an optional parameter $j = C_+/C_-$ was provided to
control different weightings of training errors on positive examples to errors on
negative examples. Therefore, we denote this method as the SVM with $j$-trick.

(Lewis et al.2004) used another method for improving SVM performance on very
imbalanced data, based on the observation that SVM learning often choses a poor
threshold (namely bias term) when the numbers of positive examples and negative
elements are very different. Hence the paper suggested that, after training the
SVM, the bias term $b$ of the classifier should be replaced by a better one, which was
obtained by a threshold selecting strategy called Scut (Yang2001). The experiments
in (Lewis et al.2004; Li and Shawe-Taylor2003) showed that Scut achieves better
performance than $j$-trick.

As discussed above, the SVMUM approach amounts to shifting the bias term $b$.
Hence, in theory it is equivalent to Scut, when one regards the latter as a heuristic
method for estimating the uneven margins parameter $\tau$. On the other hand, the
parameter $\tau$ (i.e., the ratio of the negative margin to the positive one) is more sensible and understandable than the bias term. For instance, given a classification problem with only a few positive training examples but many negative ones, we know that a small value of the margin parameter ($<1$) should be used in SVMUM, but it is much harder to determine what is a suitable value of the SVM bias (as needed by Scut). Therefore, while SVMUM is theoretically equivalent to Scut, in practice, SVMUM is better, because it is easier to find an (sub-)optimal value for the uneven margins parameter than for the bias term. This has also been backed by experimental results where SVMUM outperforms Scut on some imbalanced datasets (see (Li and Shawe-Taylor 2003)).

In addition, it has been demonstrated that SVM without the bias term is more useful than SVM with bias term on some problems like adaptive document filtering (see e.g. (Cancedda et al. 2003)). Therefore, SVMUM is better than Scut in such cases, as it introduces a margin parameter (which exists in both kinds of SVM methods), whereas the Scut method is only applicable to SVM with a bias term.

As demonstrated above, both SVMUM and Scut achieved better performance than j-trick. Therefore one remaining question is why should the uneven margins model provide higher accuracy than the j-trick one? Firstly, as illustrated above, when there are only a few positive training examples they tend to be unrepresentative of all positive examples and therefore moving the SVM hyperplane away from the positive training examples would tend to improve the generalisation capability of the SVM model. One way to achieve this is by adjusting the uneven margins parameter (or equivalently the bias term). In contrast, in the case of the 1-norm soft margin formulation of the SVM which is used in most applications, the j-trick amounts to using a larger cost parameter $C_+$, which is the upper bound of the dual form variable $\alpha_i$ for one positive training example $x_i$ (for details see (Cristianini

---

4 There is one-one correspondence between changing the uneven margins parameter and changing the bias term, as shown in the transformation formula (2).
Adapting SVM for Data Sparseness and Imbalance (and Shawe-Taylor2000)). From the SVM margin’s point of view, the j-trick method tries to increase the margin of the positive training examples in an indirect fashion by increasing the upper bound of their dual form variables. So while increasing the value of $\alpha_i$ would certainly increase the margin of the corresponding example $x_i$, increasing the upper bound of $\alpha_i$ would not necessarily result in a bigger value of $\alpha_i$, because $\alpha_i$ is determined by solving the quadratic optimal problem which has many other constraints besides the upper bound. In our view, this is the reason why j-trick was not as effective as the uneven margins model and Scut when dealing with imbalanced training data.

4 SVM Active Learning for IE

In addition to the problem with imbalanced training data, there is also the problem of obtaining sufficient training data for IE. In general, a learning algorithm derives a model from a manually annotated set of 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 of examples, which are selected from a pool of unlabelled documents, as in the applications such as the mixed-initiative text annotation (Day et al.1997) and adaptive IE (Ciravegna et al.2002).

One way to overcome this problem is to use active learning which minimises the number of labelled 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 labelled 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 unlabelled text, among which only the most informative instances need to be found and annotated. Another way for the SVM to exploit unlabelled data is the transductive SVM (see
SVM active learning is an SVM-specific algorithm (Campbell, Cristianini, and Smola2000; Schohn and Cohn2000), 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 (Tong and Koller2001), spoken language understanding (Tur, Schapire, and Hakkani-Tur2003), named entity recognition (Vlachos2004) and Japanese word segmentation (Sassano2002). These experiments have shown that SVM active learning clearly outperformed SVM with passive learning, however no attempt was made to tackle the problem of imbalanced training data at the same time.

This section explores how active learning can be combined with the SVMUM model, in order to address both problems simultaneously. The approach is based on the observation that the margin of the SVM classifier to one individual example can be used both for dealing with imbalanced data and for measuring how informative the example is for training the SVM classifier – this forms the basis for our version of the SVM active learning algorithm. In addition, we address some specific issues in the application of the SVM active learning to the natural language learning, which does not occur in other applications such as image classification and text categorisation.

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

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

which measures how close the example is to the SVM 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 in classifying the example \( X \) correctly. 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 model. Therefore, the SVM active learning algorithm is based on the margin – it selects the example(s) with the smallest margin (least confidence).

The following 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. Most of previous applications of active learning to IE selected documents as examples, but a set of tokens was used as examples in (Wu and Pottenger2005; Jones2005). Therefore, this paper presents experimental results evaluating the three types of examples, in order to identify which one is most suitable.

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

\[
    c_d = \sum_{i=1}^{m} \sum_{j=1}^{m_0} m_{ij}
\]

(4)

The document with the smallest average confidence would be selected.

The second issue is the optimal setting of parameters of the active learning algorithm. Our experiments (Section 5) evaluate 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);
- \( m_0 \) (the number of tokens chosen in one document for calculating the average confidence of the document).

The third issue is related to the combination of active learning with SVMUM. As discussed in Section 3, SVMUM is used for classification, by solving a standard SVM problem, in order to address the problem with imbalanced data. However, for the purposes of active learning, the margin for each token in the unlabelled documents is computed using the standard SVM model. That is so, because SVMUM is actually based on a standard SVM and therefore unlabelled examples which are close to the standard SVM classifier rather than the deduced SVMUM would have important effect on the next round of learning. As a matter of fact, we did use the margin computed with respect to the SVMUM in active learning and the results were clearly worse than those using the margin of the standard SVM, which verified our conjectures above.

We will discuss these three issues in more detail in the experiments presented below.

5 Experiments

We evaluated SVMUM and SVM active learning on three IE benchmark corpora covering different IE tasks – named entity recognition (CoNLL-2003) and template
Adapting SVM for Data Sparseness and Imbalance

filling (Seminars and Jobs). CoNLL-2003\(^5\) is the most recent corpus for English named entity recognition. The Seminars and Jobs corpora\(^6\) have been used to evaluate active learning techniques for IE, thus enabling a comparison with previous work: (Finn and Kushmerick2003), (Califf1998) and (Ciravegna et al.2002).

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 Jobs corpus includes 300 software related job postings and 17 slots encoding job details, such as title, salary, recruiter, computer language, application, and platform. The Seminars Corpus contains 485 seminar announcements and four slots – start time (stime), end time (etime), speaker and location of the seminar.

Unless stated otherwise, the experiments in this paper use the SVMUM parameter settings derived empirically in (Li, Bontcheva, and Cunningham2005a). 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>
<thead>
<tr>
<th></th>
<th>Context window size</th>
<th>SVM kernel</th>
<th>(\tau) (uneven margins parameter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conll03</td>
<td>4</td>
<td>quadratic</td>
<td>0.5</td>
</tr>
<tr>
<td>Seminars</td>
<td>5</td>
<td>quadratic</td>
<td>0.4</td>
</tr>
<tr>
<td>Jobs</td>
<td>3</td>
<td>linear</td>
<td>0.4</td>
</tr>
</tbody>
</table>

\(^5\) See http://cnts.uia.ac.be/conll2003/ner/

\(^6\) See http://www.isi.edu/info-agents/RISE/repository.html.
5.1 Experiments with SVMUM

Named Entity Recognition Table 2 compares the two learning algorithms: SVMUM and the standard SVM on the CONLL-2003 test set, together with the results of two systems, which participated in the CoNLL-2003 shared task: the best system (Florian et al. 2003) and the SVM-based system (Mayfield, McNamee, and Piatko 2003).

Table 2. Results on the CoNLL-2003 corpus: F-measure(%) on each entity type and 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.

<table>
<thead>
<tr>
<th>System</th>
<th>LOC</th>
<th>MISC</th>
<th>ORG</th>
<th>PER</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMUM</td>
<td>89.25</td>
<td>77.79</td>
<td>82.29</td>
<td>90.92</td>
<td>86.30</td>
</tr>
<tr>
<td>Standard SVM</td>
<td>88.86</td>
<td>77.32</td>
<td>80.16</td>
<td>88.93</td>
<td>85.05</td>
</tr>
<tr>
<td>Best one</td>
<td>91.15</td>
<td>80.44</td>
<td>84.67</td>
<td>93.85</td>
<td>88.76 ±0.7</td>
</tr>
<tr>
<td>Another SVM</td>
<td>88.77</td>
<td>74.19</td>
<td>79.00</td>
<td>90.67</td>
<td>84.67 ±1.0</td>
</tr>
</tbody>
</table>

As can be seen, our SVMUM 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 linguistic features. Therefore, in order to make a fair comparison between SVMUM and the standard SVM model, we ran experiments with the standard SVM model using the same features, as these of SVMUM. As can be seen from Table 2, even under the same experimental settings, SVMUM still outperforms the standard SVM model.

Note that our results were still lower than the best result. In fact, among the 16 participating systems of the CoNLL-03 shared task, only two systems obtained better results than our SVMUM (see (Sang and Meulder 2003)). The best system used four different machine learning algorithms and combined the results from the four classifiers in a complicated way (Florian et al. 2003). In contrast, our system
Adapting SVM for Data Sparseness and Imbalance

was only based on one learning algorithm SVM. One could expect that the result would be improved if we combine SVMUM with other learning algorithms. Another system obtaining better results than ours used only one classifier, maximum entropy. However, it adopted the so-called global features and the local features. Our system just used the local features. As showed in (Chieu and Ng2002b), the global features alone could improved the F-measures by 2-3 points.

To show the effect of the uneven margins on the SVM’s performance, Table 3 presents the results of SVMUM with several different values for the uneven margins parameters. It also shows the results in terms of Precision, Recall and F1, in order to show the effect of uneven margins on balancing Precision and Recall.

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>1.0</th>
<th>0.8</th>
<th>0.6</th>
<th>0.4</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>93.51</td>
<td>92.44</td>
<td>89.76</td>
<td>86.81</td>
<td>79.64</td>
</tr>
<tr>
<td>Recall</td>
<td>77.99</td>
<td>80.43</td>
<td>82.87</td>
<td>84.95</td>
<td>83.64</td>
</tr>
<tr>
<td>$F_1$</td>
<td>85.05</td>
<td>86.02</td>
<td>86.18</td>
<td>85.87</td>
<td>81.59</td>
</tr>
</tbody>
</table>

As we said earlier, the uneven margins parameter $\tau = 1.0$ corresponds to the standard SVM. We can see that it resulted in imbalanced Precision and Recall, i.e., high Precision and low Recall. The $\tau$ parameter allowed the system to achieve more balanced Precision and Recall — a $\tau$ less than 1 will decrease the bias term of the SVM model and hence increase Recall. For example, comparing $\tau = 0.6$ to 1.0, the Precision decreased by 4 points but the Recall increased by 5 points, and as a result, the F1 increased by 1 points. Secondly, when $\tau$ changed from 0.8 to 0.4, the Precision decreased and the Recall increased, but the $F_1$ did not changed much (within 0.5 point), showing that the $F_1$ was not very sensitive to the $\tau$ value, as long as it is not very small or big.

**Template Filling** The effect of the uneven margins parameter on SVM per-
formance was also evaluated on the jobs corpus and, in addition, SVMUM is compared to several other state-of-the-art learning systems, including the rule based systems Rapier (Califf1998), (LP)$^2$ (Ciravegna2001) and BWI (Freitag and Kushmerick2000), the statistical system HMM (Freitag and Kushmerick2000), and the double classification system (Sitter and Daelemans2003). 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 4 presents the $F_1$ measure for each slot together with the overall macro-averaged $F_1$. It should be noted that the majority of previous systems only reported per slot F-measures, without overall results. However, an overall measure is useful when comparing different systems on the same dataset, so the macro-averaged $F_1$ for these systems was computed from their per-slot $F_1$.

Table 4 presents the results of our SVMUM system (SVMUM), the standard SVM model, 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.

The results show that the overall performance of SVMUM is significantly better than the standard SVM model as well as 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, despite having obtained the best result on one of the slots. HMM was evaluated only on two slots. It achieved the best result on one slot but had significantly worse performance on the other slot. BWI obtained better results than SVMUM on three slots, but due to lack of results
Table 4. *Comparison of SVMUM against standard SVM and 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 appear in bold.

<table>
<thead>
<tr>
<th>Slot</th>
<th>SVMUM</th>
<th>SVM ($LP)^2$</th>
<th>Rapier</th>
<th>DCs</th>
<th>BWI</th>
<th>HMM</th>
<th>CRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>97.7</td>
<td>97.3</td>
<td>100</td>
<td>97</td>
<td>97</td>
<td>100</td>
<td>–</td>
</tr>
<tr>
<td>Title</td>
<td>49.6</td>
<td>47.6</td>
<td>43.9</td>
<td>40.5</td>
<td>35</td>
<td>50.1</td>
<td>57.7</td>
</tr>
<tr>
<td>Company</td>
<td>77.2</td>
<td>73.8</td>
<td>71.9</td>
<td>70.0</td>
<td>38</td>
<td><strong>78.2</strong></td>
<td>50.4</td>
</tr>
<tr>
<td>Salary</td>
<td><strong>86.5</strong></td>
<td>76.6</td>
<td>62.8</td>
<td>67.4</td>
<td>67</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Recruiter</td>
<td>78.4</td>
<td>78.2</td>
<td><strong>80.6</strong></td>
<td>68.4</td>
<td>55</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>State</td>
<td>92.8</td>
<td>91.2</td>
<td>84.7</td>
<td>90.2</td>
<td>94</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>City</td>
<td><strong>95.5</strong></td>
<td>95.2</td>
<td>93.0</td>
<td>90.4</td>
<td>91</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Country</td>
<td><strong>96.2</strong></td>
<td>95.1</td>
<td>81.0</td>
<td>93.2</td>
<td>92</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Language</td>
<td>86.9</td>
<td>86.3</td>
<td><strong>91.0</strong></td>
<td>81.8</td>
<td>33</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Platform</td>
<td>80.1</td>
<td>77.3</td>
<td><strong>80.5</strong></td>
<td>72.5</td>
<td>36</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Application</td>
<td>70.2</td>
<td>65.6</td>
<td><strong>78.4</strong></td>
<td>69.3</td>
<td>30</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Area</td>
<td>46.8</td>
<td>47.2</td>
<td><strong>53.7</strong></td>
<td>42.4</td>
<td>17</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Req-years-e</td>
<td><strong>80.8</strong></td>
<td>78.0</td>
<td>68.8</td>
<td>67.2</td>
<td>76</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Des-years-e</td>
<td>81.9</td>
<td>80.1</td>
<td>60.4</td>
<td><strong>87.5</strong></td>
<td>47</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Req-degree</td>
<td><strong>87.5</strong></td>
<td>82.2</td>
<td>84.7</td>
<td>81.5</td>
<td>45</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Des-degree</td>
<td>59.2</td>
<td>39.0</td>
<td>65.1</td>
<td><strong>72.2</strong></td>
<td>33</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Post date</td>
<td>99.2</td>
<td><strong>99.5</strong></td>
<td><strong>99.5</strong></td>
<td>99.5</td>
<td>98</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

| MA $F_1$   | **80.8 ± 0.7**| 77.1 ± 1.3 | 77.2 | 76.0 | 57.9 | –   | –   | –   |

on the other slots, it is impossible to compare the two algorithms on the entire Jobs dataset.

**Impact of Corpus Size on Performance** Our hypothesis was that the uneven margins parameter would be even more helpful on smaller training sets, because the smaller the training set, the more imbalanced it could be. For instance, each document in the jobs corpus provides typically one positive example per slot and the tokens not belonging to any slot vastly outnumber the annotated tokens.

Therefore we carried out experiments starting with a small number of training documents and gradually increased the corpus size, which simulated machine learning application in the real application scenarios such as mixed-initiative text.
annotation (Day et al. 1997) and adaptive IE (Ciravegna et al. 2002). Table 5 shows the results of the standard SVM and SVMUM on different numbers of training documents from the CoNLL-2003 and Jobs datasets, respectively. The performance of both SVMs improves consistently as more training documents are used. Moreover, the smaller the training set is, the more better the results of the SVMUM are in comparison to the standard SVM.

Table 5. The performances of the SVM system with small training sets: macro-averaged $F_1$ (%) on the CoNLL-2003 (test set) and Jobs. The SVMUM ($\tau = 0.4$) is compared to the standard SVM model with even margins ($\tau = 1$). The 95\% confidential intervals for the Jobs dataset are also presented, showing the statistical significance of the results.

<table>
<thead>
<tr>
<th>Training data</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 0.4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conll</td>
<td>56.0</td>
<td>63.8</td>
<td>67.6</td>
<td>69.4</td>
<td>71.9</td>
</tr>
<tr>
<td>Jobs</td>
<td>51.6 ±1.9</td>
<td>60.9 ±1.8</td>
<td>65.7 ±1.4</td>
<td>68.6 ±1.4</td>
<td>71.1 ±1.8</td>
</tr>
<tr>
<td>$\tau = 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conll</td>
<td>41.2</td>
<td>54.6</td>
<td>61.2</td>
<td>66.5</td>
<td>68.4</td>
</tr>
<tr>
<td>Jobs</td>
<td>47.1 ±2.4</td>
<td>56.5 ±2.2</td>
<td>61.4 ±1.9</td>
<td>65.4 ±1.4</td>
<td>68.1 ±1.5</td>
</tr>
</tbody>
</table>

5.2 Experiments with SVM Active Learning

As already discussed in Section 4, active learning first selects some examples for initial learning, then in each round 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.
Adapting SVM for Data Sparseness and Imbalance

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

5.2.1 Active Learning vs. Random Selection

The first comparison is between SVM active learning and 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 Section 5.2.2 on how these values were obtained).

Figure 2 presents the learning curves for SVM active learning and random selection on the three datasets. Both used SVMUM. The results for the standard SVM with random selection are also presented at some data points for comparison. A learning curve shows the performance trend of a system evaluated on test set when more and more labelled data are added to training set.

Firstly, 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 SVMUM performs significantly better than the standard SVM with random 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
Fig. 2. Learning curves respectively on the three datasets: the overall F1 plotted against different numbers of documents in training set. Active learning is compared with random selection for the SVMUM algorithm. The results for the standard SVM model with 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. Active and Random in the graphs refer to SVMUM with active learning and random selection, respectively, and Standard SVM refers to standard SVM with random selection.

bigger than the corresponding ones for active learning at almost all data points on all three datasets, showing that the results of active learning are more stable.

Next we compare SVM active learning to other approaches, using the seminars corpus, as the majority of previous work has been carried out on this dataset. For instance, (Finn and Kushmerick2003) evaluated several active learning techniques based on the \((LP)^2\) rule learning algorithm. Table 6 compares SVMUM active learning with the results presented in (Finn and Kushmerick2003), under similar experimental settings\(^7\).

\(^7\) Because (Finn and Kushmerick2003) presented their results in the form of graphs instead of tables, their results in Table 6 were estimated by us from their graphs.
We can see that SVMUM active learning has much better performance than the possible best results achieved by rule-based active learning discussed in (Finn and Kushmerick 2003). In fact, our active learning results are close to the optimal results, which was estimated as the upper bound on the performance of any selection strategy.

On the Jobs dataset, the SVMUM active learning results are better than the active learning method based on Rapier (Califf 1998). 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).

(Vlachos 2004) applied SVM active learning on the CoNLL-03 corpus for named entity recognition, using the same learning algorithm and similar NLP features. However, there are two main differences between his experiments and ours. The first one is that (Vlachos 2004) learned one binary SVM classifier for each entity type (to discriminate the tokens belonging to any of the entities of that type from all other tokens), while we train two binary classifiers for each entity type (see Section 2.2). The second difference is that (Vlachos 2004) used the standard SVM algorithm while we use SVMUM.

Table 7 compares our experimental results respectively using the standard SVM and SVMUM against the corresponding results of the standard SVM with and without active learning, as reported in (Vlachos 2004). First, the active learning clearly outperformed the passive learning in both cases, confirming the effectiveness of the SVM active learning. Secondly, the difference in the results of the standard SVM in our system and those reported by (Vlachos 2004) is due to the different experi-

---

Table 6. Comparisons of our results with previous best results and the upper bounds: the overall $F_1$ (%) on the Seminar corpus.

<table>
<thead>
<tr>
<th>#docs</th>
<th>SVMUM+AL</th>
<th>Finn03</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>73.0</td>
<td>61</td>
<td>77</td>
</tr>
<tr>
<td>30</td>
<td>76.1</td>
<td>66</td>
<td>79</td>
</tr>
</tbody>
</table>
mental settings, particularly in learning one versus two binary classifiers per entity type. However, the results using SVMUM in our experiments were significantly better than the results of the standard SVM in both our experiments and those in (Vlachos2004), showing the advantage of the SVMUM model. Last but not least, the best results are obtained again when the active learning is combined with the SVMUM.

Table 7. Comparison of our experimental results on the CoNLL-03 corpus with those presented in (Vlachos 2004) (which we estimated from Figure 5.5 (the min curve) in (Vlachos 2004)): Micro-averaged $F_1$ (%). “AL+SVMUM” refers to SVM active learning with an SVMUM model. “SVM” and “SVMUM” respectively refers to the standard SVM and the SVMUM using random selection. The highest score appears in bold.

<table>
<thead>
<tr>
<th>#Training docs</th>
<th>Our results</th>
<th>Results of (Vlachos2004)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>SVMUM</td>
</tr>
<tr>
<td>10</td>
<td>41.2</td>
<td>56.0</td>
</tr>
<tr>
<td>20</td>
<td>54.6</td>
<td>63.8</td>
</tr>
<tr>
<td>30</td>
<td>61.2</td>
<td>67.6</td>
</tr>
</tbody>
</table>

5.2.2 Parameters of SVM Active Learning

As discussed in Section 4, 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 (4). 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 8 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 between 3 and 5.
Table 8. 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.

<table>
<thead>
<tr>
<th>$m_0$</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seminars</td>
<td>66.2 ±1.6</td>
<td>70.4 ±1.4</td>
<td><strong>72.0 ±1.8</strong></td>
<td>69.3 ±1.7</td>
</tr>
<tr>
<td>Jobs</td>
<td>64.2 ±1.6</td>
<td><strong>65.2 ±1.2</strong></td>
<td>65.0 ±0.7</td>
<td>63.1 ±1.7</td>
</tr>
</tbody>
</table>

The second parameter is the number of initial documents randomly chosen for training, namely $n_0$. 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 9 shows the results for $n_0=1, 2, 4, \text{ and } 10$, 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 9. 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.

<table>
<thead>
<tr>
<th>$n_0$</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seminars</td>
<td>72.2 ±0.7</td>
<td><strong>73.0 ±0.9</strong></td>
<td>72.0 ±1.8</td>
<td>68.3 ±1.6</td>
</tr>
<tr>
<td>Jobs</td>
<td>64.1 ±1.2</td>
<td>65.0 ±1.1</td>
<td><strong>65.8 ±0.8</strong></td>
<td>65.2 ±1.2</td>
</tr>
</tbody>
</table>

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 at a time. On the other hand, the more documents are selected in one loop, the fewer loops the system may need in order to reach a given performance target. 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 10 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 10. 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.

<table>
<thead>
<tr>
<th></th>
<th>(n=1)</th>
<th>(n=2)</th>
<th>(n=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seminars</td>
<td>73.0 ± 1.0</td>
<td>72.8 ± 0.9</td>
<td>71.5 ± 1.2</td>
</tr>
<tr>
<td>Jobs</td>
<td>65.0 ± 1.0</td>
<td>64.2 ± 0.7</td>
<td>64.1 ± 1.0</td>
</tr>
</tbody>
</table>

5.2.3 Three Types of Samples

As discussed in Section 4, active learning for Information Extraction can select as an example in each loop either an entire document, or one or more tokens. The drawback of selecting complete documents in each loop is that the user must annotate them from beginning to end, which could be quite a substantial task if the documents are bigger. Therefore, compared to full document samples, if token or text fragment sampling can be used, then it could save a great deal of manual annotation time in each active learning loop. Consequently, this section compares the performance of token and text fragment sampling against 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.

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 need to make fewer decisions. 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 (token sampling) or 55 tokens (token with context sampling) belong to an entity of a particular type. Consequently, the annotation time for document sampling would be between 67 and 6 times more than that for token and token with context sampling, respectively.

6 Related Work

To the best of our knowledge, the problem of dealing with imbalanced training data in NLP applications has so far received limited attention. (J.Gimenez and
Marquez2003) 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 (see e.g. (Zhang and Mani2003)). (Kudoh and Matsumoto2000) used pairwise classification which trains a classifier for each pair of classes. In pairwise classification the negative and positive examples are drawn respectively from only two classes, so the training data would be much less imbalanced than in the general multi-class case. In comparison, our method of dealing with imbalanced training data is simpler, as it modifies the learning algorithm itself and thus does not require special processing of the training data or pairwise classification. The multi-class SVM is another way to deal with imbalanced data (Crammer and Singer2001; Tsochantaridis et al.2004).

Section 3.1 presents a detailed comparison of the SVMUM with other two approaches, namely j-trick and Scut for helping SVM handling imbalanced training data.

Active learning has been shown to be very useful in alleviating the need of manual annotated training data in a variety of NLP learning, such as base noun phrase chunking (Ngai and Yarowsky2000), Japanese word segmentation (Sassano2002) and statistical parsing (Hwa2004). With respect to its use for information extraction, relevant approaches were discussed in Section 5.2, but it should be noted that all but one involve non-SVM machine learning algorithms. There are several papers which studied the application of SVM active learning to NLP problems. (Sassano2002) investigated SVM active learning for Japanese word segmentation and had the same basic findings as ours, namely that SVM active learning can significantly improve performance. This paper also demonstrated 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 the focus of the experiments in this paper.

Use of SVM active learning for named entity recognition and NP chunking was studied in (Vlachos2004), especially different approaches for selecting unlabelled examples based on the confidence score of the SVM classifier. Vlachos also used the CoNLL-2003 share task data, which enables a direct comparison between his results and ours. In a nutshell, Table 7 showed that combining active learning with SVMUM achieves the best results.

7 Conclusions

This paper investigated two techniques for enhancing the standard SVM model to deal with imbalanced training data and the difficulty in obtaining human-annotated examples, which are two problems that frequently arise in NLP tasks. In comparison to other methods dealing with these problems, our methods are effective and efficient, which are valuable features for real-world applications.

The paper presents a new approach towards dealing with imbalanced training data by introducing the uneven margins parameter in the SVMUM model. We also investigated SVM active learning and the different strategies that can be used in order to reduce the required human input – full document, single token, and tokens in context. Both SVMUM and SVM active learning only require solving the standard binary SVMs and using some simple transformations. Hence they are simpler and more efficient than some other SVM-based methods such as multi-class SVM and transductive SVM, because the latter need implement and solve some more complicated optimisation problems than the standard SVM. On the other hand, our methods are more effective than other methods such as those transforming the training data. Therefore our methods are appealing in real-world applications many of which tend to have hundreds of classes and thousands or millions of instances.

SVMUM and SVM active learning were evaluated independently of each other
and also in combination, by applying them to two information extraction tasks, i.e., named entity recognition and template filling. The results demonstrate clearly that SVMUM performs better than the standard SVM model and SVM active learning also achieves better performance than random sampling. Moreover, when the two approaches are combined, the system outperforms other state-of-the-art methods evaluated on the same benchmark IE datasets.

Since the SVMUM model can be obtained from a standard SVM classifier by changing the bias term, it can be implemented easily using any publicly available SVM package. Based on our results and previous work, we believe that SVMUM could bring similar performance improvements on NLP tasks other than IE. Similarly, the SVM active learning presented in this paper is applicable to most NLP problems.

One avenue of future work is to investigate enhancing the multi-class SVM (see e.g. (Tsochantaridis et al.2004)) to deal with imbalanced training data and the transductive SVM (TSVM, see e.g. (Collobert et al.2006)) to exploit unlabelled data, and compare and/or combine them with the methods discussed in this paper.

Another future work is to make further investigation on the types of examples used in active learning. As discussed in Section 5.2.3, the document-level example resulted in a good learning curve, while the token-level example took less time for manual annotation in each learning round. The combination of the two types of examples, e.g. using the token-level example in several consecutive learning rounds then using document-level example in one or two subsequent rounds, would probably strike a good balance between system accuracy and annotation time.

We are also currently experimenting with applying SVMUM and active learning to other NLP problems, particularly on opinion analysis and Chinese word segmentation.

**Acknowledgments:** This research was partially supported by the EU-
funded MUSING (http://www.musing.eu/) and SEKT projects (http://www.sekt-project.com).

References


Adapting SVM for Data Sparseness and Imbalance


Adapting SVM for Data Sparseness and Imbalance


